Constraints-based Verification of Parameterized Cryptographic Protocols.

Najah Chridi, Mathieu Turuani, Michaël Rusinowitch

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Constraints-based Verification of Parameterized Cryptographic Protocols

Najah Chridi — Mathieu Turuani — Michael Rusinowitch

N° 6712

November 2008

Thème SYM

ISSN 0249-6399 ISRN INRIA/RR--6712--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
NANCY - GRAND EST

Constraints-based Verification of Parameterized Cryptographic Protocols

Najah Chridi , Mathieu Turuani , Michael Rusinowitch

Thème SYM — Systèmes symboliques
Équipes-Projets Cassis

Rapport de recherche n° 6712 — November 2008 — 51 pages

Abstract: Cryptographic protocols are crucial for securing electronic transactions. The confidence in these protocols can be increased by the formal analysis of their security properties. Although many works have been dedicated to standard protocols like Needham-Schroder very few address the more challenging class of group protocols. We present a synchronous model for group protocols, that generalizes standard protocol models by permitting unbounded lists inside messages. In this extended model we propose a correct and complete set of inference rules for checking security properties in presence of an active intruder for the class of Well-Tagged protocols. We prove that the application of these rules on a constraint system terminates and that the normal form obtained can be checked for satisfiability. Therefore, we present here a decision procedure for this class.

Key-words: Cryptographic protocols, Inference system, Group protocols, rewriting, security, verification, constraints

This work was supported by AVANTSSAR, FP7-ICT-2007-1 Project No.216471, and SeComMaNet, PRST MISN Project 2007-2013.

Centre de recherche INRIA Nancy – Grand Est
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy
Téléphone : +33 3 83 59 30 00 — Télécopie : +33 3 83 27 83 19

Vérification de protocoles cryptographiques paramétrés à base de résolution de contraintes

Résumé : Les protocoles cryptographiques sont cruciaux pour la sécurité des transactions éléctroniques. La confiance en ces protocoles peut être améliorée par une analyse formelle de leurs propriétés de sécurité. Bien que de nombreux travaux aient été consacrés aux protocoles classiques tels que celui de Needham-Schroder, très peu s'intéressent la classe prometteuse des protocoles de groupe. Nous présentons un modèle synchrone de protocoles de groupe qui généralise les modèles classiques en incluant des listes non bornées dans les messages. Dans ce modèle étendu, nous proposons un ensemble correct et complet de règles d'inférence pour la vérification de propriétés de sécurité en présence d'un intrus actif pour une classe de protocoles bien taggés. Nous prouvons que l'application de ces règles pour un système de contraintes termine et que la forme normale obtenue peut Ã ªtre testée pour la satisfiabilité. Ainsi, nous présentons ici une procédure de décision pour cette classe.

Mots-clés : protocoles cryptographiques, système d'inférence, protocoles de groupe, réécriture, sécurité, vérification, contraintes

1 Introduction

Cryptographic protocols are crucial for securing electronic transactions. They rely on cryptographic functions to ensure security properties such as secrecy or authentication. The confidence in these protocols can be increased by a formal analysis in order to verify that the security properties are met at least at the logical level, that is, even when abstracting from the cryptographic functions and considering messages as first-order terms. Verification at the logical level is nevertheless a non-trivial task since cryptographic protocols are infinite state systems and for instance the set of potential messages that can be generated by an intruder is unbounded. Recently numerous works have been dedicated to the design of automated verification tools for cryptographic protocols. Such tools are often based on model-checking, modal logics, equational reasoning, and resolution theorem-proving (see e.g., [Wei99, AC02, MT07]). Checking whether a protocol is flawed in the abstract Dolev Yao model ([DY83]) can often be reduced to a constraint solving problem in a term algebra (modulo an equational theory). This constraint-based approach has proved to be quite effective on standard benchmarks and has also permitted the discovery of new flaws in several protocols (see e.g., [BMV03]).

However to our knowledge it has never been applied to the more challenging group protocols. In fact very few formal verification results are available for such protocols. The difficulty relates to the fact that group protocols may perform an arbitrary number of steps since the group of communicating agents is a priori unbounded. This allows one to encode easily undecidable problems.

In this work we present a synchronous model for group protocols, that can also be viewed as an extension of standard protocol models ([RT03]) to handle messages containing possibly unbounded lists. In this extended model we propose a correct and complete set of inference rules that allows one to check security properties in presence of an active intruder, for a class of so-called *Well-Tagged* protocols. We show that relaxing the conditions on Well-Tagged protocols leads immediately to undecidability. Our inference rules generalize the ones that are implemented in several tools for a bounded number of sessions and fixed size lists such as Cl-ATSE ([Tur06]). In particular when applied to protocols with standard pair operator our inference system provides us with a decision procedure.

Related work Several works have considered protocols with unbounded number of participants and recursive steps. The formal analysis of such protocols goes up with [Pau97] who studied the Recursive Authentication (RA) protocol of [JD97] for an unbounded number of participants using Isabelle/HOL theorem prover ([Pau96]). However if the protocol is defective there is no automatic mechanism to find the attack.

The validation of group protocols has been investigated in the CLIQUES project ([SWT98]), based on group Diffie-Hellman (A-GDH) protocols. Several analysis methods have been applied in this project, from manual to automatic ones. An interesting result in this area has been obtained by [PQ03] who found several attacks on the CLIQUES suite and have shown that it is impossible to design a correct authentication group key agreement protocol built on A-GDH for a number of participants greater than three ([PQ04]). Recently, [KMT08] have developped an automata-based approximation technique to analyse this class of protocols and check the absence of flaw in presence of a **passive intruder**. [MS01] have adapted the NRL protocol analyser, which is based on a combination of model checking and theorem-proving techniques, to handle the GDOI's protocols. Although Diffie-Hellman exponentiation has been encoded in the tool, it was not able to rediscover Pereira-Quisquater attacks on the CLIQUES suite ([Mea00]). [SB04] have used Coral system to analyse an improved version of the multicast group key management protocol by Tanaka and Sato ([TJ03]). Two serious attacks have been found on this protocol. Coral has also discovered other attacks on Asokan-Ginzboorg protocol([AG00]) and Iolus protocol([Mit97]).

Some works have focused on the modelling of recursive computations performed by some participants (such as a server) in group protocols. [KW04] introduce tree transducers to model recursion and to allow the protocol participants to output structured messages. This work gives a decision algorithm for secrecy in the case of atomic keys and bounded message size in the Dolev Yao setting. However messages cannot be tested for equality without losing decidability. Similarly using composed keys or adding equational theories for XOR or Diffie-Hellman exponentiation in their model leads to undecidability. [Tru05] introduces a class of Horn clauses to model the recursive behavior of participants. In this model protocol participants may receive messages of unbounded sizes, send multiple messages in a single step, compare and store messages. He gives a decision procedure to check whether protocols in this model satisfy secrecy properties. His algorithm is in NEXPTIME and is based on the derivation of an exponential bound on the size of minimal attacks. Hence this nice result is rather of theoretical flavour and is not suitable for an implementation. Only atomic keys are allowed for encryption. Moreover, Truderung's model cannot model some computations such as list mapping or functional symbol mapping. Note that non-atomic keys can be handled by our verification procedure (to be presented in

the following sections). [KT07] extend this model to handle XOR operator. Security can then be decided for a class of recursive protocols where principals are forbidden to XOR several messages (depending on messages) received from the network. [KRT07] extend the Truderung model to model freshness of nonces and keys more accurately.

The class of protocols that we study admits tagged messages. Tagging basically avoids some unifications between messages that could be exploited for attacks. Several works on protocols have considered tagging techniques on messages as ours in order to enforce decidability. But these works ([BP03, RS03]) do not consider group protocols, or protocol with unbounded lists. Moreover in our case tagging is limited to messages that contains indexed variables, that is variables to be instanciated by items of unbounded lists. The other messages do not need to be tagged in our analysis.

Organization of the paper. We introduce in Section 2 our protocol model. We define attacks and show that their detection is undecidable. This motivates us to introduce the class of Well-Tagged protocols with autonomous keys for which we prove decidability. In Section 3, we introduce auxiliary predicates and their semantics. They are meant to express message constructibility (from intruder knowledge) and they are used to build constraint system whose satisfiability is equivalent to the existence of attacks on a protocol. In Section 4, we introduce a set of simplification rules to reduce these constraints. In Section 5, we give an algorithm for applying these rules on a constraint system modelling a protocol security problem. Completeness and correctness are proved in Section 6. Proof of termination for Protocols without $mpair(,)$'s is given in Section 7. Proof of termination for Well-Tagged Protocols with Autonomous Keys is given in Section 8. In Section 9, we prove satisfiability. The analysis of the the Asokan-Ginzboorg protocol by our Rules and our verification algorithm is detailed in Section 10.

1.1 Motivating Example: Synchronous Group Protocols

As a motivating and running example, we introduce the Asokan-Ginzboorg group protocol which is an application level protocol. Let the group be of size $n+1$ for $n \geq 1$. The protocol describes the establishment of a session key between a leader (a_n) and a random number n of participants (a_i where $1 \leq i \leq n$). Indeed, the leader starts the execution of the protocol by sending the key of encoding (e). As a response, each participant generates a symmetric key (r_i) and a contribution to the group key (s_i) and sends them encrypted under the key e. The group key would be $f(s_1, \ldots, s_{n+1})$.

$$
\begin{array}{lll}
1.\ a_{n+1} \rightarrow \text{ALL} : & \langle a_{n+1}, \{e\}_p \rangle & \\
2.\ a_i \rightarrow a_{n+1} \quad : & \langle a_i, \{\langle r_i, s_i \rangle\}_e \rangle & 1 \leq i \leq n \\
3.\ a_{n+1} \rightarrow a_i \quad : & \{\langle s_1, .., s_{n+1} \rangle\}_{r_i} & 1 \leq i \leq n \\
4.\ a_i \rightarrow a_{n+1} \quad : & \langle a_i, \{\langle s_i, h(s_1, \ldots, s_{n+1}) \rangle\}_K \rangle & \text{some } i,\ K = f(s_1, \ldots, s_{n+1})
\end{array}
$$

Since here, modulo index renaming, all members of the group have identical actions, we are going to abstract them in one agent S. Thus, S is a simulator for agents $a_1, \ldots, a_n$. Agent L simulates the leader a_{n+1}. This way, we obtain below a protocol with a fixed number of steps to the expense of introducing a variadic list constructor $mpair(\_, \_)$. This protocol is called *synchronous protocol*. Note that all parametric lists have the same length n. They are nested as follows:

$$
\begin{array}{llll}
1.\ L & \longrightarrow & S & : mpair(t, \langle l, \{e\}_p \rangle) \\
2.\ S & \longrightarrow & L & : mpair(i, \langle a_i, \{\langle r_i, s_i \rangle\}_e \rangle) \\
3.\ L & \longrightarrow & S & : mpair(i, \{\langle mpair(j, s_j), s' \rangle\}_{r_i}) \\
4.\ S & \longrightarrow & L & : mpair(i, \langle a_i, \{\langle s_i, h(\langle mpair(k, s_k), s' \rangle) \rangle\}_{f(\langle mpair(k,s_k),s' \rangle)} \rangle)
\end{array}
$$

We assume that the leader L has in initial knowledge his name l (a_{n+1} in the initial version), the fresh public key e, the symmetric key p and two hash functions f and h ($f, h \in H$). Its contribution to the group key is noted s' (s_{n+1} in the initial version). The simulator S knows the symmetric key p, the two hash functions f and h and $\forall i$ the identity a_i, the contribution (nonce) s_i and the public key r_i.

2 The Protocol Model

We extend the protocol model [RT03] in order to deal with parametric lists (whose length is the parameter). They are constructed with a new operator denoted by $mpair(\_, \_)$. The intuition is that a *mpair* message is equivalent to a list of messages built with the same pattern.

2.1 Names, Operators and Messages

Let $\mathcal{X}$ be a set of variables represented by capital letters. Let $\mathcal{I}$ be a countable set of index variables. Let $\overrightarrow{\mathcal{X}}$ be a set of symbols represented by overarrowed capital letters, disjoint from $\mathcal{X}$. Let $\mathcal{X}_\mathcal{I} = \{Y_i \text{ s.t. } \overrightarrow{Y} \in \overrightarrow{\mathcal{X}} \text{ and } i \in \mathcal{I}\}$ be a countable set of (indexed) variables. Similarly, let $\mathcal{C}$ and $\overrightarrow{\mathcal{C}}$ be (disjoint) sets of symbols, represented by (overarrowed) lower case letters, and let $\mathcal{C}_\mathcal{I} = \{c_i \text{ s.t. } c \in \overrightarrow{\mathcal{C}} \text{ and } i \in \mathcal{I}\}$. Elements in $\mathcal{C}$ and $\mathcal{C}_\mathcal{I}$ are called constants. In this paper, terms can be (optionally) tagged by an index. That is, let $\overrightarrow{e} \in \overrightarrow{\mathcal{C}}$ be a symbol that we reserve for tagging operations only. Then a term is an element of $\mathcal{T}$ in the following language:

$$\begin{array}{lcl} \mathcal{T}_s & = & \{\mathcal{T}\}^p_\mathcal{T} \mid \{\mathcal{T}\}^s_\mathcal{T} \mid h(\mathcal{T}) \mid \langle \mathcal{T}, \mathcal{T} \rangle \mid mpair(\mathcal{I}, \mathcal{T}) \mid \mathcal{X} \mid \mathcal{X}_\mathcal{I} \mid \mathcal{C} \mid \mathcal{C}_\mathcal{I} \backslash \{e_i \mid i \in \mathcal{I}\} \\ \mathcal{T} & = & [e_i, \mathcal{T}_s] \mid \mathcal{T}_s \qquad \text{with } i \in \mathcal{I} \end{array}$$

where $\mathcal{T}_s$ is by definition the set of untagged terms. The operators $\{\_\}^p_\_$ and $\{\_\}^s_\_$ represent asymmetric and symmetric encryptions respectively, $\langle \_, \_ \rangle$ is a pairing operator, h is a hash function and $mpair(i,t)$ is a symbolic representation of a list (or tuple) of terms, built from the common pattern t by iterating i along integers. The translation function defined in Section 2.4 gives the semantics of this operator. We denote by signature $\mathcal{G}$ the set of operators in $\mathcal{T}_s$. To simplify the syntax, in the following we will write t^i instead of $[e_i, t]$, and call it a *tagged term*. We also omit the tag i of a term t^i, whenever the tag i is not relevant to the discussion. We denote by $\mathcal{T}_g$ the set of ground terms, i.e. any term $t \in \mathcal{T}$ with no variable in $\mathcal{X}$ or $\mathcal{X}_\mathcal{I}$ and no *mpair* symbol. Ground terms will be used to describe messages that are circulated in a protocol run. Given a term t we denote by $Var(t)$ (resp. $Cons(t)$) the set of variables (resp. constants) occuring in t. We denote by $Atoms(t)$ the set $Var(t) \cup Cons(t)$.

In order to represent a list of terms we iterate the pairing operator $\langle \_, \_ \rangle$. For instance to represent a, b, c, d we can use the term $\langle a, \langle b, \langle c, d \rangle \rangle \rangle$ and we shall write this term in a shorthand: $\langle a, b, c, d \rangle$. However we do not assume any associativity property of pairing.

A substitution σ assigns terms to variables. A ground substitution assigns ground terms to variables. The application of σ to a term t is written $t\sigma$. These notations are extended to sets of terms E in a standard way: $E\sigma = \{t\sigma | t \in E\}$. The set of subterms of t is denoted by $Subterm(t)$. It is defined recursively as follows: If t is a variable or a constant then $Subterm(t) = \{t\}$. If $t = f(t_1, .., t_n)$ or $t = f(t_1, .., t_n)^i$ with $t \in \mathcal{G}$, then $Subterm(t) = \{t\} \cup \cup_{i=1}^n Subterm(t_i)$. Note that u is *not* considered as a subterm of u^i. We denote by $\leq$ the subterm relation on $\mathcal{T}$. We define the relation $\leqslant_m$ over $\mathcal{T} \times \mathcal{T}$ as the smallest reflexive and transitive relation such that if $t = f(t_1, .., t_n)$ or $t = f(t_1, .., t_n)^j$ with $f \neq mpair$, then for all $i = 1, \ldots, m$ we have $t_i \leqslant_m f(t_1, .., t_m)$. Note that $t \leqslant_m u$ implies $t \leq u$.

Finally, we define two kind of Index-operations: replacements, used in the inference rules over constraints, and substitutions, used to define the solutions of constraints (See Section 3).

Definition 1 (Index-Replacement δ, Index-Substitution τ)**.** *An Index-Replacement δ (resp. Index-Substitution τ) is an application from $\mathcal{I}$ to $\mathcal{I}$ (resp. to non-negative integers) that is extended to indexed variables and constants with $\delta(X_i) = X_{\delta(i)}$ and $\delta(c_i) = c_{\delta(i)}$ (resp. $\tau(X_i) = X_{\tau(i)}$ and $\tau(c_i) = c_{\tau(i)}$) and extended to terms and sets of terms in the natural way.*

We will use the notations $\delta_{i,j}$ (resp. $\tau_{i,j}$) to denote the replacement (resp. substitution) of $i \in \mathcal{I}$ by $j \in \mathcal{I}$ (resp. $j \in \mathbb{N}$). We also use $\delta^k_{i,j}$ to denote the replacement of $i \in \mathcal{I}$ by $j \in \mathcal{I}$ and the other indexes apart from i by $k \in \mathcal{I}$. We extend this notation to sets with $\delta^k_{Q,j}$ for some set $Q \subseteq \mathcal{I}$.
We define the set of indexes occuring in a term as follows:

Definition 2 (Term Indexes)**.** *Given a term $t \in \mathcal{T}$, we denote by $Var_\mathcal{I}(t)$ the set of indexes in t, recursively defined as follows:*

$$\begin{array}{ll} Var_\mathcal{I}(mpair(i,t)) = Var_\mathcal{I}(X) = Var_\mathcal{I}(c) = \emptyset & \text{with } X \in \mathcal{X} \text{ and } c \in \mathcal{C} \\ Var_\mathcal{I}(X_i) = Var_\mathcal{I}(c_i) = \{i\} & \text{with } X_i \in \mathcal{X}_\mathcal{I} \text{ and } c_i \in \mathcal{C}_\mathcal{I} \\ Var_\mathcal{I}(f(t_1, .., t_n)) = Var_\mathcal{I}(t_1) \cup .. \cup Var_\mathcal{I}(t_n) & \text{otherwise} \\ Var_\mathcal{I}(t^i) = Var_\mathcal{I}(t) \cup \{i\} & \end{array}$$

Moreover, we define also the set of indexes of variables in $u \in \mathcal{T}$ as

$$Var^\mathcal{X}_\mathcal{I}(u) = Var_\mathcal{I}(u) \cap Var_\mathcal{I}((Subterm(u) \cap \mathcal{X}_\mathcal{I}))$$

2.2 Protocol Specification

A protocol is given by a set of principals and a finite list of steps for each. We associate to each principal A a partially ordered finite set $(W_A, <_{W_A})$ steps $R_\iota \Rightarrow S_\iota$ where R_ι is an expected message and S_ι his reply. *Init* and *End* are fixed messages used to initiate and close a protocol session. Our notion of correct execution of a protocol session (or *protocol run*) follows [RT03].

Example 1. *We return to our running example: the "synchronous version" of AG protocol with $n+1$ participants (See Section 1.1). The specification of this protocol is given below:*

$$
\begin{array}{lll}
(L,1) & \mathit{Init} & \Rightarrow \quad \mathit{mpair}(t, \langle l, \{e\}_p \rangle) \\
(S,1) & \mathit{mpair}(i, \langle L, \{E_i\}_p \rangle) & \Rightarrow \quad \mathit{mpair}(j, \langle a_j, \{\langle r_j, s_j \rangle\}_{E_j} \rangle) \\
(L,2) & \mathit{mpair}(k, \langle a_k, \{\langle R_k, S_k \rangle\}_e \rangle) & \Rightarrow \quad \mathit{mpair}(m, \{\langle \mathit{mpair}(o, S_o), s' \rangle\}_{R_m}) \\
(S,2) & \quad \mathit{mpair}(q, \{\langle \mathit{mpair}(u, s_u), S' \rangle\}_{r_q}) & \Rightarrow \\
 & \mathit{mpair}(w, \langle a_w, \{\langle s_w, H(\langle \mathit{mpair}(y, s_y), S' \rangle)\rangle\}_{F(\langle \mathit{mpair}(y,s_y),S' \rangle)} \rangle) & \\
(L,3) & \mathit{mpair}(x, \langle a_x, \{\langle S_x, H(\langle \mathit{mpair}(z, S_z), s' \rangle)\rangle\}_{F(\langle \mathit{mpair}(z,S_z),s' \rangle)} \rangle) \Rightarrow \mathit{End} &
\end{array}
$$

The ordering on steps is: $W_L = 1, 2, 3$, $W_S = 1, 2$ with $1 <_{W_L} 2$, $2 <_{W_L} 3$, and $1 <_{W_S} 2$.

2.3 Intruder

We follow the intruder model of Dolev and Yao [DY83]. The actions of the intruder are simulated by a sequence of rewrite rules on sets of messages. These rules are defined as follows. We note $\longrightarrow^*_{DY}$ their reflexive and transitive closure.

$L_d(\langle a_1, \ldots, a_n \rangle) : \langle a_1, \ldots, a_n \rangle \rightarrow a_1, \ldots, a_n, \langle a_1, \ldots, a_n \rangle$	$L_c(\langle a_1, \ldots, a_n \rangle) : a_1, \ldots, a_n \rightarrow a_1, \ldots, a_n, \langle a_1, \ldots, a_n \rangle$
$L_d(\{a\}^p_K) : \{a\}^p_K, K^{-1} \rightarrow \{a\}^p_K, K^{-1}, a$	$L_c(\{a\}^p_K) : a, K \rightarrow a, K, \{a\}^p_K$
$L_d(\{a\}^s_K) : \{a\}^s_K, b \rightarrow \{a\}^s_K, b, a$	$L_c(\{a\}^s_K) : a, b \rightarrow a, b, \{a\}^s_K$
$L_d(t^i) : t^i \rightarrow t$	$L_c(t^i) : t \rightarrow t^i$ for any $i \in \mathcal{I}$

2.4 Attacks

We recall from [RT03] the notion of derivation D of goal t from E, denoted by $D_t(E)$. We define a Non-Redundant Derivation as follows:

Definition 3 (Non-Redundant Derivation)**.** *Given a derivation $D = E_0 \longrightarrow_{L_1} \ldots \longrightarrow_{L_l} E_l$ with goal u. D is a non redundant derivation if $\forall i \, \forall t \in E_i$, if $L_c(t) \in D$ then $\nexists L_d(\_) \in D$ that generates t, and if $\exists L_d(\_) \in D$ generating t then $L_c(t) \notin D$. We denote by NRD the set of non-redundant derivations.*

Remark 1. *For each derivation $D_t(E)$, there exists a non redundant derivation $D'_t(E)$. Indeed, D' is obtained by elimination of $L_c(t)$ if $L_d(\_) \in D$ where $L_d(\_)$ generates t and by eliminating each $L_d(\_) \in D$ that generates t if $L_c(t) \in D$.*

We define a predicate Dy. This predicate checks whether a message can be constructed by the intruder from some known messages.

Definition 4 (Dy, Dy_c and Dy_d)**.** *Let E be a set of ground terms, $\mathcal{K}$ be a set of ground terms and t be a ground term such that there exists $D \in NRD$ with goal t without using any term of $\mathcal{K}$ as a key for decryption. Then, we say that t is forged from E and we denote it by $t \in Dy(E, \mathcal{K})$. Moreover, if $D = D'.L_c(t)$ then $t \in Dy_c(E, \mathcal{K})$, otherwise $t \in Dy_d(E, \mathcal{K})$.*

We interpret the $\mathit{mpair}(\_, \_)$ operator in the standard Dolev-Yao signature by defining a translation function that replaces any *mpair* by a sequence of *pair* applications. The number of such applications is given as a parameter e to the translation function. The integer represents the common length of lists of terms represented by any $\mathit{mpair}(,)$.

Definition 5 (Translation of terms)**.** *Let $\mathcal{T}_{DY}$ be the set of terms without any $\mathit{mpair}(,)$. Given some integer e, the function $^{-e}$ from $\mathcal{T}$ to $\mathcal{T}_{DY}$ is defined as follows:*

$$\overline{\mathit{mpair}(i,t)}^e = \langle \overline{\tau_{i,1}(t)}^e, .., \overline{\tau_{i,v}(t)}^e \rangle \quad \text{and} \quad \overline{f(s_1, .., s_k)}^e = f(\overline{s_1}^e, .., \overline{s_k}^e), \text{ for any } f \neq \mathit{mpair}$$

We can now define attacks on protocols in our model, based on Dy predicate.

Definition 6 (Attack)**.** *Given a protocol $P = \{R'_\imath \Rightarrow S'_\imath | \imath \in J\}$, a secret Sec and assuming the intruder has as initial knowledge S_0, an attack is described by a ground substitution σ, an Index-Substitution τ, an integer e, and a correct execution order $\pi : J \longrightarrow 1, \ldots, k$ s.t. $\forall i = 1, \ldots, k$, we have:*

$$\overline{R_i\tau\sigma}^e \in Dy(\overline{S_0}^e, \overline{S_1}^e\tau\sigma, \ldots, \overline{S_{i-1}}^e\tau\sigma, \emptyset)$$
$$\overline{Sec}^e \in Dy(\overline{S_0}^e, \overline{S_1}^e\tau\sigma, \ldots, \overline{S_k}^e\tau\sigma, \emptyset)$$

where $R_i = R'_{\pi^{-1}(i)}$ and $S_i = S'_{\pi^{-1}(i)}$.

2.5 Undecidability and Well-Tagged Protocols

Unfortunately, the insecurity problem (i.e. the existence of an attack) is undecidable in the general case. This can be shown by encoding Post Correspondance Problem (PCP) with two letters. Note that this requires only atomic keys.

Definition 7 (PCP protocol)**.** *Let $J = \{(\alpha_1, \beta_2), .., (\alpha_p, \beta_p)\}$ be an instance of PCP on the alphabet $\{a, b\}$. We define the protocol specification $P(J)$ coding J as the following, with $\mathcal{C} = \{a, b, 0, t, u\}$, $\mathcal{X} = \{Z\}$, $\overrightarrow{\mathcal{X}} = \left\{\overrightarrow{A}, \overrightarrow{B}, \overrightarrow{X}, \overrightarrow{Y}\right\}$ and only one honest participant :*

1.	$Init$	$\Rightarrow$	a, b, 0, $\{\langle 0, 0\rangle\}_t$
2.	$mpair(i, \langle A_i, B_i\rangle)$	$\Rightarrow$	$mpair(i, \{\langle A_i, B_i\rangle\}_t)$
3.	$mpair(i, \{\langle X_i, Y_i\rangle\}_t)$	$\Rightarrow$	$mpair(i, \{\langle \alpha_1(X_i), \beta_1(Y_i)\rangle\}_u), \ldots$
			$mpair(i, \{\langle \alpha_p(X_i), \beta_p(Y_i)\rangle\}_u)$
4.	$mpair(i, \{\langle A_i, B_i\rangle\}_u), \{\langle Z, Z\rangle\}_u$	$\Rightarrow$	Sec

At Step 1, we provide the alphabet as constants to the intruder, as well as a termination symbol 0. At Step 2, the intruder is asked to construct a list (or $mpair$) of pairs of words over a, b. The following of the protocol will consist in testing if each of these pairs can be obtained from an other one increased with one of the PCP words (α_j, β_j). Since the initial pair of empty words represented by $\{\langle 0, 0\rangle\}_t$, cannot be obtained this way, it is provided to the intruder separately at Step 1. We fix the intruder choice at step 2 by encrypting it with key u. Then, at Step 3 the intruder selects some of the pairs of words encrypted by t that he got at step 1 and 2, and receives each of them back extended with one of the pairs of words of the PCP instance and encrypted by u. We expect the intruder to select all of the pairs he has chosen at Step 2 minus the longest one, replaced by the pair of empty words. Finally, at Step 4 we perform two verifications: first, we test that for any pair of words chosen at Step 2, there exists an extended pair of words received at Step 3, i.e. by recursion that each pair of terms chosen at Step 2 is a concatenation of words of J, the instance of PCP from which we build this protocol; second, we test that one of the extended pair of words is a solution to PCP problem, i.e. a pair of identical words.

Theorem 1. *An instance J of PCP has a solution iff $P(J)$ has an attack on Sec.*

The proof of this theorem follows from Lemmas 1 and 2.

Lemma 1. *If for some integer n there is a run of $P(J)$, then J has a solution.*

Proof. According to the protocol definition, all we need to do is to backtrack recursively the creation of extended pairs at Step 2 and 3: for any term $\{\langle a, b\rangle\}_t$ known by the intruder, including $\{\langle A_j, B_j\rangle\}_u$ for any $j \in 1..n$ or $\{\langle Z, Z\rangle\}_u$, there exists $i \in 1..n$ and $k \in 1..p$ such that $a = \alpha_k(X_i)$ and $b = \beta_k(Y_i)$. Therefore, either $X_i = Y_i = 0$, or there exists $i' \in 1..n$ such that $a = \alpha_k(A_{i'})$ *and* $b = \beta_k(B_{i'})$, and the intruder knows $\{\langle A_{i'}, B_{i'}\rangle\}_t$. By iteration on $A_{i'}$and $B_{i'}$, starting from $\{\langle Z, Z\rangle\}_u$, and since $|\langle a, b\rangle| > |\langle A_{i'}, B_{i'}\rangle|$, it appears that there exists a list $[j_1, .., j_r]$ of indexes in $1..p$ such that $\alpha_{j_1}(\alpha_{j_2}(..\alpha_{j_r}(0)..)) = \beta_{j_1}(\beta_{j_2}(..\beta_{j_r}(0)..))$, i.e. $\alpha_{j_1}..\alpha_{j_r} = \beta_{j_1}..\beta_{j_r}$. □

Lemma 2. *If there is a solution to J, there exists n such that $P(J)$ admits a run.*

Proof. Let $[j_1, .., j_r]$ be a list of indexes in $1..p$ such that $\alpha_{j_1}..\alpha_{j_r} = \beta_{j_1}..\beta_{j_r}$. We chose $n = r$. We also choose the following values for variables A_i, B_i :

$$\forall i \in 1..n-1,\ \sigma(A_i) = \alpha_{j_i}(\sigma(A_{i+1})) \text{ and } \sigma(B_i) = \beta_{j_i}(\sigma(B_{i+1}))$$
$$\sigma(A_n) = \alpha_{j_n}(0) \text{ and } \sigma(B_n) = \beta_{j_n}(0)$$

This is the set of pairs of words chosen by the intruder at step 2. Then, at step 3 we chose :

$$\forall i \in 2..n,\ \sigma(X_i) = \sigma(A_i) \text{ and } \sigma(Y_i) = \sigma(B_i)$$
$$\sigma(X_1) = \sigma(Y_1) = 0$$

i.e. we keep all the pairs of words chosen at step 2 except that we replace the longest (final) one, A_1, B_1 by the pair of empty words. We can now easily pass the two tests of step 4, since :

$$\forall i \in 1..n-1,\ \exists k \in 1..p,\ \exists j \in 1..n \text{ s.t. } \sigma(A_i) = \alpha_k(\sigma(X_j)) \text{ and } \sigma(A_i) = \beta_k(\sigma(X_j))$$
$$\sigma(A_n) = \alpha_{j_n}(0) \text{ and } \sigma(B_n) = \beta_{j_n}(0)$$
$$\sigma(A_1) = \alpha_{j_1}(..\alpha_{j_r}(0)..) = \beta_{j_1}(..\beta_{j_r}(0)..) = \sigma(B_1)$$

□

Consequently, finding an attack to PCP with two letters is no more difficult than finding a run in a parameterized protocol as defined in this paper. Since finding a run is no more difficult than finding an attack (the secret can be released at the end), it follows that the insecurity problem of parameterized protocol without further restrictions is undecidable.

We will therefore introduce the class of Well-Tagged protocols for which decidability is expected. To do this, we first introduce the notion of autonomy:

Definition 8 (Autonomy)**.** *A term $mpair(i,u)$ is autonomous when $Var_{\mathcal{I}}(u) \subseteq \{i\}$. A term $t \in T_{DY}$ is autonomous if $\#Var_{\mathcal{I}}(t) \leq 1$ and $\forall t' < t$, t' is autonomous. A protocol $\mathcal{P} = \{R_\imath \Rightarrow S_\imath | \imath \in J\}$ is autonomous iff for all $\imath \in J$, $R_\imath$ and $S_\imath$ are autonomous and $Var_{\mathcal{I}}(R_\imath) = \emptyset$ and $Var_{\mathcal{I}}(S_\imath) = \emptyset$.*

For instance, the term $t = mpair(i, mpair(j, \{a_i\}_{c_j}))$ is not autonomous. We remark that the autonomy property alone is not enough to guarantee decidability, since the PCP protocol of Definition 7 is autonomous.

Definition 9. *(Well-Tagged protocols)*
A protocol $\mathcal{P} = \{R_\imath \Rightarrow S_\imath | \imath \in J\}$ is Well-Tagged iff:

1. *$\forall \imath \in J$, $\forall X_i \in \mathcal{X}_{\mathcal{I}} \cap Subterm(R_\imath) \cap \bigcup_{\imath' < \imath} Subterm(R_{\imath'})$, X_i is tagged;*
2. *$\forall \imath \in J$, $\forall X_i \in \mathcal{X}_{\mathcal{I}} \cap Subterm(S_\imath)$, X_i is tagged;*
3. *$\forall \imath \in J$, $\forall t = f(s_1,..,s_k) \in Subterm(R_\imath \cup S_\imath)$ with $f \neq mpair$, if $\exists j = 1..k$ s.t. s_j is tagged, then t is tagged too;*
4. *$\forall \imath \in J$, $\forall t \in Subterm(R_\imath)$ tagged, $\forall X_i \leq t$ where $X_i \in \mathcal{X}_{\mathcal{I}}$, X_i is tagged.*
5. *$\mathcal{P}$ is autonomous.*

In this definition, Conditions 1 and 2 state that any indexed variable of the protocol must be tagged, except for its first occurrence w.r.t. the partial step ordering. Moreover, Condition 3 (when combined with Conditions 1 and 2) states that, for any subterm t of the protocol, if an indexed variable is accessible from t by decompositions without opening any *mpair*, then t must be tagged. Note that as a consequence of mpair autonomy, an indexed variable X_i can only appear tagged by its index (as in ${X_i}^i$) or untagged. Condition 4 states that every indexed variable subterm of a tagged term of $R_\imath$ is tagged.

The idea underlying the tagging of variables is to add enough information on terms in *mpair* so that the protocol cannot be used to test or guarantee relations between elements of the same *mpair*, such as $\forall i = 2..n$, $\exists i' = 1..n$ s.t. $X_i = f(X_{i'})$. This is precisely the kind of relations that the encoding of PCP is able to exploit. Thus, adding tags to the PCP-encoding protocol will generate a new protocol that cannot be run.

We introduce the notion of protocols with autonomous keys that will be used to prove the termination for those protocols.

Definition 10. *(Protocol with Autonoumous Keys)*
A protocol $\mathcal{P} = \{R_\imath \Rightarrow S_\imath | \imath \in J\}$ is called with autonomous keys iff $\forall \imath \in J$ $t \in Subterm(R_\imath \cup S_\imath)$ s.t $t = \{u\}_v$, $Var_{\mathcal{I}}(v) = \emptyset$.

3 Constraints for Protocol Verification

We will use a symbolic constraint system to represent all runs of a protocol given a step ordering. This system uses (universal or existential) quantifiers on index variables and includes an (implicit) universal quantification on the number n of elements in any *mpair*. Before defining our constraint system, some basic notions have to be introduced. For terms s, s' (resp. sets of terms E, E') we note $s \sim s'$ (resp. $E \sim E'$) if they are equal once we erase their tags.

Definition 11 (Relation $\leqslant_E^L$ for accessible subterms)**.** *We consider a relation $\_\leqslant_{\_}^{\_}\_$ on $\mathcal{T} \times 2^{\mathcal{T}} \times 2^{\mathcal{T}} \times \mathcal{T}$. We write $s \leqslant_E^L t$ for s, t terms in $\mathcal{T}$ and E and L finite subsets of $\mathcal{T}$. Note that this can be used for $\mathcal{T}_s$ too. This relation is defined as the smallest relation such that:*

$t \leqslant_\emptyset^\emptyset t$	$\forall t \in \mathcal{T}$
$s \leqslant_E^L t$	*iff* $s' \leqslant_{E'}^{L'} t'$ *where* $s \sim s', t \sim t', E \sim E', L \sim L'$
If $\{m\}_k^p \leqslant_E^L t$	*then* $m \leqslant_{E,k^{-1}}^{L'} t$ *where* $L' = L \cup \{\{m\}_k^p\}$
If $\{m\}_b^s \leqslant_E^L t$	*then* $m \leqslant_{E,b}^{L'} t$ *where* $L' = L \cup \{\{m\}_b^s\}$
If $\langle t_1, \ldots, t_n \rangle \leqslant_E^L t$	*then* $\forall i \leq n, t_i \leqslant_E^{L'} t$ *where* $L' = L \cup \{\langle t_1, \ldots, t_n \rangle\}$
If $m \leqslant_{E'}^L t$ *and* $E' \subset E$	*then* $m \leqslant_E^L t$

We note $u \leqslant_E t$ when $u \leqslant_E^L t$ for some L. Remark that by construction, $u \leqslant_E t$ implies $u \in Subterm(t)$. We say that u is a subterm of t that is accessible, i.e. can be obtained from t by decompositions using keys in E. For simplicity, we note $\leqslant_{b_1,..,b_k}$ instead of $\leqslant_{\{b_1,..,b_k\}}$. We define also the set of strict accessible terms by $s <_F t$ (resp. $s <_F^L t$) if $s \leqslant_F t$ (resp. $s \leqslant_F^L t$) and $s \neq t$. Given $t \leqslant_F^L u$ or $t <_F^L u$), we call length of $t \leqslant_F^L u$ or (resp $t <_F^L u$) the number of elements of L.

Before defining constraint systems, we need to introduce the environment and elementary constraints.

Definition 12 (Environment)**.** *We call an* environment *a finite set of equalities $X = u$ whose left-hand sides are variables ($X \in \mathcal{X} \cup \mathcal{X}_\mathcal{I}$). We usually denote it by $\mathcal{E}$.*

Definition 13 (Elementary Constraint)**.** *An elementary constraint is an expression $(t \in Forge(E, \mathcal{K}))$, $(t = t')$, $(t \in Sub(t', E, \mathcal{E}, \mathcal{K}))$, $(t \in Sub_d(t', E, \mathcal{E}, \mathcal{K}))$ or $(t \in Forge_c(E, \mathcal{K}))$ with $t, t' \in \mathcal{T}$, $E \subset \mathcal{T}$ and an environment $\mathcal{E}$.*

An elementary constraint represents a basic relation on terms: $t \in Forge(E, \mathcal{K})$ if the term t is derivable from the knowledge E without using any element from $\mathcal{K}$ as a key for decryption; $t \in Forge_c(E, \mathcal{K})$ if t is derived by composition; $t \in Sub(t', E, \mathcal{E}, \mathcal{K})$ if t is an accessible subterm from t' with knowledge E with none of keys of the intermediate terms between t and t' in $\mathcal{K}$, and this modulo replacements using equations of $\mathcal{E}$; $t \in Sub_d(t', E, \mathcal{E}, \mathcal{K})$ if t is accessible by decomposition of t', also modulo replacements using $\mathcal{E}$; and $t = t'$ if t and t' are equal.

Definition 14 (Negative Constraint)**.** *A negative constraint is an expression $(\forall i\ X_m \neq u)$ or $(X_m \notin Forge_c(E, \mathcal{K}))$ with $X_m \in \mathcal{X}_\mathcal{I}$, $u \in \mathcal{T}$, $E, E' \subset \mathcal{T}$ and $i \in Var_\mathcal{I}(u)$.*

The set of solutions of a constraint S, denoted by $[\![S]\!]_\tau^e$ where e is a value of n and τ is an Index-Substitution is a set of ground substitutions to be defined in the following. We define $\mathcal{G}S$ to be the set of all ground substitutions.

Definition 15 (Solutions of an Elementary Constraint)**.**

$$
\begin{array}{lcl}
[\![t = t']\!]_\tau^e & = & \{\sigma \in \mathcal{G}S \mid \overline{t}^e \tau\sigma = \overline{t'}^e \tau\sigma\} \\
[\![t \in Forge(E, \mathcal{K})]\!]_\tau^e & = & \{\sigma \in \mathcal{G}S \mid \overline{t}^e \tau\sigma \in Dy(\overline{E}^e \tau\sigma, \overline{\mathcal{K}}^e \tau\sigma)\} \\
[\![t \in Forge_c(E, \mathcal{K})]\!]_\tau^e & = & \{\sigma \in \mathcal{G}S \mid \overline{t}^e \tau\sigma \in Dy_c(\overline{E}^e \tau\sigma, \overline{\mathcal{K}}^e \tau\sigma)\}
\end{array}
$$

$$
\begin{array}{lcl}
[\![t \in Sub(w, E, \mathcal{E}, \mathcal{K})]\!]_\tau^e & = & \{\sigma \in \mathcal{G}S \mid \exists u\ \exists F, L\ s.t.\ u \leqslant_F^L \overline{w}^e \tau,\ F\sigma \subseteq Dy(\overline{E}^e \tau\sigma, \overline{\mathcal{K}}^e \tau\sigma), \\
& & F\sigma \cap \overline{\mathcal{K}}^e \tau\sigma = \emptyset,\ \text{and either } u\sigma = \overline{t}^e \tau\sigma \text{ or } \exists v, \delta, k,\ \tau'\ s.t \\
& & \left\{ \begin{array}{l}
\left\{ \begin{array}{l}
\text{either } u \in \mathcal{X},\ (u = v) \in \mathcal{E},\ \delta = \emptyset,\ \text{and } \tau' = \tau \\
\text{or } \exists \overrightarrow{Z} \in \overrightarrow{\mathcal{X}},\ i, j \in \mathcal{I}\ s.t\ u = Z_i \tau',\ (Z_j = v) \in \mathcal{E}, \\
\quad \delta = \delta_{j,i}^k, \tau \subseteq \tau' \text{ and } \mathcal{D}om(\tau') = \mathcal{D}om(\tau) \cup \{k, i\}
\end{array} \right. \\
k, i \notin Var_\mathcal{I}(\{t, w, \mathcal{E}\}), u\sigma \notin Dy_c(\overline{E}^e \tau\sigma, \overline{\mathcal{K}}^e \tau\sigma) \\
u\sigma = \overline{v}^e \delta\tau'\sigma,\ \text{and } \sigma \in [\![t \in Sub(v\delta, E, \mathcal{E}, \mathcal{K})]\!]_{\tau'}^e
\end{array} \right\}
\end{array}
$$

$[\![t \in Sub_d(w, E, \mathcal{E}, \mathcal{K})]\!]_\tau^e$ is defined in a similar way as $[\![t \in Sub(w, E, \mathcal{E}, \mathcal{K})]\!]_\tau^e$ with the difference that for the first case (when $u\sigma = \overline{t}^e \tau\sigma$), we have $u <_F^L \overline{w}^e \tau$.

Definition 16 (Solutions of a Negative Constraint)**.**

$$\begin{array}{lcl} [\![X_m \notin Forge_c(E,\mathcal{K})]\!]^e_\tau & = & \mathcal{GS} \setminus [\![X_m \in Forge_c(E,\mathcal{K})]\!]^e_\tau \\ [\![(\forall i\ X_m \neq u)]\!]^e_\tau & = & \mathcal{GS} \setminus \bigcup_{x=1\ldots e} [\![X_m = u]\!]^e_{[i \leftarrow x],\tau} \end{array}$$

We describe our constraint system by blocks in the following way:

Definition 17 (Constraint System)**.** *First, we define a constraints block B as a conjunction of constraints together with an environment $\mathcal{E}$:*

$$B = (ctr_1 \wedge \ldots \wedge ctr_l, \mathcal{E})$$

We will sometimes handle blocks as set of elementary or negative constraints for ease of notations. For instance we write $c \in B$ to express that the elementary constraint c is a conjunct of B.

We can now define the constraint system that we will use to represent protocol runs. Given two finite lists of index variables $Q = i_1, \ldots, i_k$ and $R = j_1, \ldots, j_l$, we write the quantifier prefix $\forall i_1 \cdots \forall i_k \exists j_1 \cdots \exists j_l$ in short: $\forall Q\ \exists R$.
A constraint system, denoted by S, is a disjunction of blocks with a quantifier prefix:

$$S \quad = \quad \forall Q\ \exists R\ (B_1 \vee \ldots \vee B_p)$$

Now, we define the set of solutions of the constraint system as follows:

Definition 18 (Solutions of the Constraint System)**.** *Consider a constraint system S, B_i, for $i = 1 \ldots p$ (blocks of S) and $ctr_{i,j}$, for $j = 1 \ldots l_i$ (constraints of the block B_i) given in Definition 17. The set of solutions of a constraint system CS is defined inductively using the following cases:*

$$\begin{array}{lcllcl} [\![\forall i\ S]\!]^e_\tau & = & \cap_{x=1,\ldots,e} [\![S]\!]^e_{[i \leftarrow x],\tau} & [\![S]\!]^e_\tau & = & \bigcup_{i=1\ldots p} [\![B_i]\!]^e_\tau \\ [\![\exists i\ S]\!]^e_\tau & = & \cup_{x=1,\ldots,e} [\![S]\!]^e_{[i \leftarrow x],\tau} & [\![B_i]\!]^e_\tau & = & \bigcap_{j=1\ldots l_i} [\![ctr_{i,j}]\!]^e_\tau \end{array}$$

The idea will be to use a constraint system based on blocks to represent all possible ways the intruder can construct a list of terms represented by an *mpair*. Roughly, there will be one block in the system for each way. Note also that blocks are extended to admit labeled constraints:

Notation 1. *(labels of constraints)*
A constraint ctr may be equiped with a label $(ctr)^m$ or $(ctr)^{sm}$ or $(ctr)^f$ to denote respectively a master constraint or a submaster constraint or a final constraint. The tow first labels allow us to keep track of the "official" formal value of some indexed or non-indexed variable. For example, we will prove that we have exactly one master constraint for every indexed variable in each block, and we will use master or sub master constraints to instanciate variables when needed; The third label will be used to prevent any further rewriting on some constraint. We introduce also the notation $(ctr)^\bullet$ to refer to labeled or non labeled constraint. The solutions of labeled constraints are the solutions of the constraints obtained by removing labels.

To simplify the use of (sub)master constraints, we group them into sets:

Definition 19 (Set of (sub)master constraints)**.** *Let $S = \forall Q \exists R\ B_1 \vee .. \vee B_p$ be a constraint system, $Y \in \mathcal{X}$, $W \subseteq \mathcal{X}$ and $\overrightarrow{X} \in \overrightarrow{\mathcal{X}}$. We define $\mathcal{M}(S, \overrightarrow{X}) = \{ctr \,|\, \exists i\ (ctr)^m \in B_i$ and $\exists j \in Q$ s.t $ctr = (X_j \in Forge_c(E,\mathcal{K}))$ or $ctr = (X_j = u)\}$ and $\forall i\ \mathcal{SM}(B_i, W) = \{ctr \,|\, (ctr)^{sm} \in B_i,\ Y \in W$ and $ctr = (Y = u)\}$. Also, $\mathcal{SM}(B_i, Y) = \mathcal{SM}(B_i, \{Y\})$.*

When S is clear from the context we omit it in $\mathcal{M}(S, \overrightarrow{X})$ and write simply $\mathcal{M}(\overrightarrow{X})$.

4 Normalisation of a Constraint System

In this section we present the rules applied in the normalization function over constraint systems. After applying a rule to a constraint system, the result is put in disjunctive normal form and existential quantifiers are moved up to the prefix of the system using first order logic. We introduce some definitions used to define our rules.

First, we extend the blocks of constraints to include an history, that is, a sorted list of rules that have occured at some step in this block:

Definition 20. *(Block History)*
Any block $B = ctr_1 \wedge ..ctr_p$ is equiped with a sorted list of rules $Hist = [r_1, \ldots]$ named the history of the block B denoted by $Hist(B)$. $Hist(B)(i)$ denotes the i^{th} rule of the history. The block is then denoted by $B = (ctr_1 \wedge ..ctr_p, \mathcal{E}, Hist)$, and constraint systems are extended to blocks with history.

Any rule application will implicitly update the history of the block in which it is applied. Therefore, in what follows, we will not write the history ($Hist$) of the block each time it is not relevant for the discussion.

We also introduce the notion of *anteriority* between index variables in the same block that will be used in the definition of Rule 25 and in proofs.

Definition 21 (anteriority of indexes)**.** *Let $B = (ctr_1 \wedge ..ctr_p, \mathcal{E}, Hist)$ be a block.*
Let i and j be two index variables.
We define the function $ant(,)$ from $\mathcal{I} \times \mathcal{I}$ to $\mathcal{I}$ such that
$ant(i,j) = i$ if $\exists \imath$ s.t $i \in post(Hist(B)(\imath))$ and $\forall \imath' \leq \imath\ j \notin post(Hist(B)(\imath'))$.

We then define the function *awake* that allows to remove the label for dull constraints. This is used when a master constraint is introduced in the block or a master constraint of type $Forge$ is changed to a new *Equality* master constraint.

Definition 22. *(Function awake) We define the function awake as follows:*

$$awake((ctr)^d) = ctr$$
$$awake((ctr)^\bullet) = (ctr)^\bullet \text{ for } (ctr)^\bullet \neq (ctr)^d$$

The function awake is extended in a natural way to constraints blocks and constraints system:
Let $B = ctr_1 \wedge \ldots \wedge ctr_m$ be a constraints block and $S = B_1 \vee \ldots \vee B_l$ be a constraints system. Then:

$$awake(B) = awake(ctr_1) \wedge \ldots \wedge awake(ctr_m)$$
$$awake(S) = awake(B_1) \vee \ldots \vee awake(B_l)$$

Finally, we define the dependency graph of a block that will be used to define the notion of accessibility of variables used in Rule 3.

Definition 23 (Dependency graph of a block)**.** *We define the dependency graph $\mathcal{G}_B$ of a block B to be the graph where $\mathcal{X} \cup \mathcal{X}_\mathcal{I}$ is the set of nodes and we have an oriented edge from X to Y iff there is $(X = u) \in B$ with $Y < u$. We write $X \sqsubset Y$ when there exists a path in $\mathcal{G}_B$ from X to Y. We also note $X \sqsubset_l Y$ if $X \sqsubset Y$ and l is the minimal length of a path from X to Y.*

Now, we present our rules system. They are organized in six groups $G_1, \ldots, G_6$.

G_1: Group of prioritary rules

G_1 aims at maintaining syntactic properties over a constraint system. Any rule in this group is applied eagerly with priority higher than any other rule applicable from another group. Moreover the rules in this group are given in strictly decreasing priority order.

$$t = X \quad \longrightarrow \quad X = t \text{ where } X \in \mathcal{X} \cup \mathcal{X}_\mathcal{I} \text{ and } t \notin \mathcal{X} \cup \mathcal{X}_\mathcal{I} \tag{1}$$

$$(X = u)^{sm} \wedge (Y = X) \quad \longrightarrow \quad (X = u)^{sm} \wedge (Y = u) \text{ for } X, Y \in \mathcal{X} \tag{2}$$

$$X = u \quad \longrightarrow \quad \bot \text{ if there is } Y < u \text{ s.t } Y \sqsubset X \text{ for } X, Y \in \mathcal{X} \cup \mathcal{X}_\mathcal{I} \tag{3}$$

$$\begin{aligned} \forall Q.i\ \exists R\ S \vee (ctr \wedge B) \quad &\longrightarrow \quad \forall Q.i\ \exists R\ awake(S) \vee ((ctr)^m \wedge awake(B)) \\ \text{where} \quad ctr \in \ &\{X_i \in Forge_c(E, \mathcal{K}), X_i = u\} \text{ and } B \cap \mathcal{M}(\overrightarrow{X}) = \emptyset \end{aligned} \tag{4}$$

$$\begin{aligned} \forall Q.i.j\ \exists R\ S \vee ((X_i \in Forge_c(E, \mathcal{K}))^m \wedge X_j = u \wedge B) \quad &\longrightarrow \quad \forall Q.i.j\ \exists R\ awake(S) \\ \vee (X_i \in Forge_c(E, \mathcal{K}) \wedge (X_j = u)^m \quad &\wedge awake(B)) \end{aligned} \tag{5}$$

$$(Y_j = Z)^{\bullet} \wedge X = u \quad \longrightarrow \quad (Y_j = Z)^{\bullet} \wedge X = u\lambda \text{ where } Y_j < u \text{ and } \lambda = [Y_j \leftarrow Z] \tag{6}$$

$$B \wedge X = u \quad \longrightarrow \quad B \wedge X = u\lambda \wedge Y_j = Z \text{ where } Y_j < u, \text{ and } \lambda = [Y_j \leftarrow Z] \text{ if for all } Z', E' \text{ we have } (Y_j = Z') \notin B \tag{7}$$

$$B \wedge X = u \quad \longrightarrow \quad B \wedge (X = u)^{sm} \text{ where } \mathcal{SM}(B, X) = \emptyset \tag{8}$$

Some rules handle labelling of master constraints by adding new labels or transfering existing ones. Indeed, Rule 4 labels master constraints for a vector variable $\overrightarrow{X} \in \overrightarrow{\mathcal{X}}$ in a block B when B does not contain yet a master constraint for $\overrightarrow{X}$. We note that Rule 4 is applied only once for each variable vector in a block. Rule 5 transfers master constraints labels from forge constraints to equality ones whenever possible. Intuitively, an equality constraint being more precise than a forge one, has to be favoured. Note that the index itself, i.e. i or j in Rule 5 is not relevant.

Other rules format constraints in order to get preferably variables on their left hand-side (Rule 1), or replace indexed variables by non-indexed ones (Rules 6 and 7).

Finally, Rule 8 manages submaster constraints by ensuring that if some block contains at least one equality fixing a value for a variable X, then exactly one of them is labelled as submaster constraint for X.

G_2: Group of $Forge$ reduction rules

G_2 aims at enumerating all possible ways a term can be built by the intruder. Here, we consider a constraints block $(B', \mathcal{E})$.

$$t \in Forge(E, \mathcal{K}) \quad \longrightarrow \quad t \in Forge_c(E, \mathcal{K}) \vee \bigvee_{w \in E} t \in Sub(w, E, \mathcal{E}, \mathcal{K}) \tag{9}$$

$$\langle t_1, \ldots, t_m \rangle \in Forge_c(E, \mathcal{K}) \quad \longrightarrow \quad \bigwedge_{i=1\ldots m} t_i \in Forge(E, \mathcal{K}) \tag{10}$$

$$\{t\}_b \in Forge_c(E, \mathcal{K}) \quad \longrightarrow \quad (b \in Forge(E, \mathcal{K}) \wedge t \in Forge(E, \mathcal{K})) \tag{11}$$

$$h(t) \in Forge_c(E, \mathcal{K}) \quad \longrightarrow \quad t \in Forge(E, \mathcal{K})) \tag{12}$$

$$\begin{array}{rll} \forall Q\ \exists R\ S \vee (B \wedge mpair(k,t) \in Forge_c(E,\mathcal{K})) & \longrightarrow & \\ \forall Q\ \exists R\ S \vee (B \wedge t\delta \in Forge(E,\mathcal{K})) & \text{if} & Hy13 \\ \forall Q.k'\ \exists R\ S \vee (B \wedge t\delta_{k,k'} \in Forge(E,\mathcal{K})) & & \text{otherwise where } k' \text{ fresh} \\ \text{and } Hy13 = ((mpair(k,t) \in Forge_c(E,\mathcal{K}) & \longrightarrow & t\delta \in Forge(E,\mathcal{K})) \in Hist(B)) \end{array} \tag{13}$$

$$c \in Forge_c(E, \mathcal{K}) \quad \longrightarrow \quad \bot, \text{ for } c \in \mathcal{C} \cup \mathcal{C}_{\mathcal{I}} \tag{14}$$

Rule 9 is a generic rule that illustrates the two possible ways for forging a term t: either by composing or by decomposing one of the knowledge. The other rules enumerate all possible ways a term can be composed by the Intruder: we have exactly one rule for decomposing each kind of operator in the signature $\mathcal{G}$ (Rules 10 to 13). In particular, Rule 13 operates for the *mpair* operator where *mpair* autonomy is used to justify the quantification.

G_3: Group of Sub reduction rules

G_3 is similar to G_2, but they decompose Intruder knowledge.

$$t \in Sub(u, E, \mathcal{E}, \mathcal{K}) \quad \longrightarrow \quad (t = u) \text{ if } u = \{v\}_b \text{ and } b \in \mathcal{K} \quad (t = u) \vee (t \in Sub_d(u, E, \mathcal{E}, \mathcal{K})) \text{ otherwise} \tag{15}$$

$$t \in Sub_d(\langle t_1, \ldots, t_m \rangle, E, \mathcal{E}, \mathcal{K}) \longrightarrow \bigvee_{i=1\ldots m} (t \in Sub(t_i, E, \mathcal{E}, \mathcal{K})) \quad (16)$$

$$t \in Sub_d(\{u\}_b^s, E, \mathcal{E}, \mathcal{K}) \longrightarrow t \in Sub(u, E, \mathcal{E}, \mathcal{K}) \wedge b \in Forge(E, \mathcal{K} \cup \{b\}) \quad (17)$$

$$t \in Sub_d(\{u\}_K^p, E, \mathcal{E}, \mathcal{K}) \longrightarrow t \in Sub(u, E, \mathcal{E}, \mathcal{K}) \wedge K^{-1} \in Forge(E, \mathcal{K} \cup \{K\}) \quad (18)$$

$$t \in Sub_d(mpair(k,u), E, \mathcal{E}, \mathcal{K}) \longrightarrow \begin{array}{l}(t \in Sub(u\delta, E, \mathcal{E}, \mathcal{K})) \text{ if } Hy19 \\ \exists k' \ (t \in Sub(u\delta_{k,k'}, E, \mathcal{E}, \mathcal{K})) \text{ otherwise}\end{array} \quad (19)$$

where k' fresh and $Hy19 = ((t \in Sub_d(mpair(k,u), E, \mathcal{E}, \mathcal{K}) \longrightarrow t \in Sub(u\delta, E, \mathcal{E}, \mathcal{K})) \in Hist(B))$

$$t \in Sub_d(c, E, \mathcal{E}, \mathcal{K}) \longrightarrow \bot, \text{ for } c \in \mathcal{C} \cup \mathcal{C}_{\mathcal{I}} \quad (20)$$

Rule 15 is a generic rule that follows precisely the intruder deduction rules: a term t is an accessible subterm of u iff $t = u$ or there exists a direct subterm u' of u, derivable from u, with t being an accessible subterm of u'. Therefore, there exists exactly one rule for decomposing each kind of operator in $\mathcal{G}$ (apart from variables and constants).

G_4: Group of simplification rules for equalities

G_4 encodes unification algorithm for terms in our system, and thus, the resolution of equality constraints. Let us note $\hat{t}$ the root symbol of a term t.

$$c = c \longrightarrow \top \text{ where } c \in \mathcal{C} \cup \mathcal{C}_{\mathcal{I}} \quad (21)$$

$$t = t' \longrightarrow \bot \text{ where } \{\hat{t}, \hat{t'}\} \subset \mathcal{C} \cup \mathcal{C}_{\mathcal{I}} \cup \mathcal{G} \text{ and } \hat{t} \neq \hat{t'} \quad (22)$$

$$f(u_1, .., u_m) = f(w_1, .., w_m) \longrightarrow \bigwedge_{i=1\ldots m} u_i = w_i \text{ where } f \in \{\{\}^p, \{\}^s, \langle\rangle, [], h\} \quad (23)$$

$$\begin{array}{rcl}\forall Q \ \exists R \ S \vee (B \wedge (mpair(k,u) = mpair(l,w))) & \longrightarrow & \\ \forall Q \ \exists R \ S \vee (B \wedge (u\delta = w\delta_{l,k}\delta)) & & if \ Hy24 \\ \forall Q.k' \ \exists R \ S \vee (B \wedge (u\delta_{k,k'} = w\delta_{l,k'})) & & \text{otherwise}\end{array} \quad (24)$$

where k' fresh and $Hy24 = (((mpair(k,u) = mpair(l,w)) \longrightarrow (u\delta = w\delta_{l,k}\delta)) \in Hist(B))$

$$\begin{array}{rcl}\forall Q \ \exists R \ S \vee (B \wedge c_j = c_i) & \longrightarrow & \forall Q \ \exists R \ S \vee (B\delta) \text{ where } c_i, c_j \in \mathcal{C}_{\mathcal{I}} \\ & & \text{and } \delta = \delta_{i,j} \text{ if } j \in Q \\ & & \text{and } \delta = \delta_{j,i} \text{ if } i \in Q \\ & & \text{and } \delta = \delta_{\{i,j\},ant(i,j)} \text{ otherwise}\end{array} \quad (25)$$

These rules simply consist in testing recursively the compatibility of each top operator in each term. Therefore, the only equality constraints remaining after an iteration of these rules are those assigning a value to a variable, i.e. $X = u$ with $X \in \mathcal{X} \cup \mathcal{X}_{\mathcal{I}}$.

G_5: Group of interleaving rules inside a block

These rules aim at replacing variables by their value, inside one block. These rules manage the interaction between two constraints in the same block. Rules in G_5 have priority on rules in G_6. Moreover, in G_5, Rule 32 has weaker priority than other rules in the same group.

$$\begin{array}{rcl} B \wedge (X_i = u)^{\bullet} \wedge (X_i = v)^{\bullet} & \longrightarrow & B \wedge (X_i = u)^{\bullet} \wedge (X_i = v)^{\bullet} \wedge u = v \\ \text{if } (B \wedge (X_i = u)^{\bullet} \wedge (X_i = v)^{\bullet} & \longrightarrow & B \wedge (X_i = u)^{\bullet} \wedge (X_i = v)^{\bullet} \wedge u = v) \notin Hist(B) \end{array} \quad (26)$$

$$(X = u)^{sm} \wedge (X = v) \longrightarrow (X = u)^{sm} \wedge (u = v) \quad (27)$$

$$(X_i = u)^{\bullet} \wedge X_i \in Forge_c(E', \mathcal{K}) \longrightarrow (X_i = u)^{\bullet} \wedge u \in Forge_c(E', \mathcal{K}) \quad (28)$$

$$(X = u)^{sm} \wedge X \in Forge_c(E', \mathcal{K}) \longrightarrow (X = u)^{sm} \wedge u \in Forge_c(E', \mathcal{K}) \quad (29)$$

$$(A \in Forge_c(E, \mathcal{K}))^{\bullet} \wedge A \in Forge_c(E', \mathcal{K}) \longrightarrow (A \in Forge_c(E, \mathcal{K}))^{\bullet} \quad \text{where} \quad E \subseteq E' \text{ and } A \in \mathcal{X} \cup \mathcal{X}_{\mathcal{I}} \quad (30)$$

$$(X = w)^{sm} \wedge t \in Sub_d(X, E', \mathcal{E}, \mathcal{K}) \wedge X \notin Forge_c(E, \mathcal{K}) \longrightarrow (X = w)^{sm} \wedge t \in Sub_d(w, E', \mathcal{E}, \mathcal{K}) \quad \text{where} \quad (X = w)^{sm} \in \mathcal{E} \quad (31)$$

$$A \in Forge_c(E, \mathcal{K}) \wedge t \in Sub_d(A, E', \mathcal{E}, \mathcal{K}) \longrightarrow \bot \text{ where } A \in \mathcal{X} \cup \mathcal{X}_{\mathcal{I}} \quad (32)$$

Rule 32 says that it is not necessary to decompose a variable. While a bit more complex than expected, our semantics of $Sub_d(,,,)$ has been defined to prevent useless actions like this, thus ensuring the validity of G_6.

G_6: Group of interleaving rules between different blocks

G_6 generalizes G_5 by allowing variable replacements from one block to an other one. Therefore, these rules are the ones that define interactions between blocks, and solve constraints between multiple indexes of the same variable in a formal way.

Given some vector variable $\overrightarrow{X} \in \overrightarrow{\mathcal{X}}$, we assume that $\mathcal{M}(\overrightarrow{X}) = \{X_{i_o} = u_o\}_{o=1..p} \cup \{X_{j_r} \in Forge_c(E'_r, K_r)\}_{r=1..q}$. Then the rewriting rules in this group for constraints over $\overrightarrow{X}$ are the following, with $\delta_o = \delta_{Q,m}^{k'_o}$, $k'_o \in \mathcal{I}$ a fresh index variable :

$$t \in Sub_d(X_m, E', \mathcal{E}, \mathcal{K}) \longrightarrow \bigvee_{(X_i = u) \in \mathcal{E}} \exists k'\ t \in Sub_d(u\delta, E', \mathcal{E}, \mathcal{K}) \wedge (X_m = u\delta)^f \wedge X_m \notin Forge_c(E', \mathcal{K}) \text{with} \, \delta \, = \, \delta_{i,m}^{k'} \quad (33)$$

$$X_m \in Forge_c(E', \mathcal{K}) \longrightarrow ((X_m \in Forge_c(E', \mathcal{K}))^d \wedge \bigwedge_{o=1...p} (\forall k'_0\ X_m \neq u_o\delta_o)) \vee \bigvee_{o=1...p} \exists k'_0\ u_o\delta_o \in Forge_c(E', \mathcal{K}) \wedge (X_m = u_o\delta_o)^f \quad (34)$$

$$X_m = v \longrightarrow \bigvee_{r=1...q} (v \in Forge_c(E'_r, \mathcal{K}_r)) \wedge (X_m = v)^d \wedge \bigwedge_{o=1...p} (\forall k'_0\ X_m \neq u_o\delta_o) \vee \bigvee_{o=1...p} \exists k'_o\ (u_o\delta_o = v) \wedge (X_m = u_o\delta_o)^f \quad (35)$$

The structure of these rules is essentially the same as for interleaving in only one block: Given a constraint containing a variable X_i that must be replaced by its value, we enumerate a finite number of "candidate" terms representing all possible values of this variable according to the whole constraint system. These values are provided by master constraints. For instance, in Rule 33, only the case where master constraints are *equality* ones are taken into account since $Forge$ ones leads to $\bot$. This rule adds an (extra) equality representing the master constraint it used. It adds also negative constraints to eliminate the case of $Forge$ master constraints.

Let B the block that we focus on for Rules 33, 34 and 35. For Rule 33, if $(t \in Sub_d(X_m, E, \mathcal{E}, \mathcal{K}) \longrightarrow \exists k'\ t \in Sub_d(u\delta, E', \mathcal{E}, \mathcal{K}) \wedge (X_m = u\delta)^f \wedge X_m \notin Forge_c(E', \mathcal{K})) \in Hist(B)$, then we preserve the same index k' for $u\delta$. That is, we generate the same Sub constraint that was generated before and which belongs to $Hist(B)$. Otherwise, i.e. the constraint $(t \in Sub_d(X_m, E, \mathcal{E}, \mathcal{K}))$ was not treated with Rule 33 by the interleaving with the master constraint $(X_i = u)$, then k' will be a fresh index.
The same reasoning is valid for Rule 34. Indeed, if the constraint $(X_m \in Forge_c(E', \mathcal{K}))$ was not already treated with Rule 34 by the interleaving with the master constraint $(X_{i_o} = u_o)$, then the index k'_o is fresh. Otherwise, we preserve the same index that was generated before by Rule 34 for the constraint $(X_m \in Forge_c(E', \mathcal{K}))$ considering the master constraint $(X_{i_o} = u_o)$ and which belongs to $Hist(B)$.
We reason similarly for Rule 35. Indeed, if the constraint $(X_m = v)$ was already treated by Rule 35 by the interleaving with a master constraint $(X_{i_o} = u_o)$ and then $(X_m = v \longrightarrow \exists k'_o\ (u_o\delta_o = v) \wedge (X_m = u_o\delta_o)^f) \in Hist(B)$, we preserve the same index variable as the one generated before. Otherwise, k'_o will be a fresh index.

Simplification rules with Tagging We have defined our constraints simplification for untagged terms. Nevertheless, these rules deal also with tagged terms following the definition of our signature. Indeed, for decomposition in $Forge$ or Sub constraints, our rules behave similarly with tagged terms as for untagged terms. Moreover, for equality constraint, the constraint $(X_i)^i = u$ leads to $\bot$ when u is untagged, since $(X_i)^i = [e_i, X_i]$. Besides, for replacement in $Forge$ or Sub constraints, tagged variables behave as 'special' variable. For example, given a constraint $(t \in Sub({X_i}^i, E, \mathcal{E}, \mathcal{K}))$, we search for master constraints for the vector $\overrightarrow{X}$. Then, assuming $(X_o = u)^m$ is one of them, the replacement result would $(t \in Sub((u\delta)^i, E, \mathcal{E}, \mathcal{K}))$ for this master constraint. The same reasoning is valid for a $Forge$ constraint.

Definition 24 (Solved Constraint)**.** *A solved constraint is of type:* $(X_i \in Forge_c(E, \mathcal{K}))^\bullet$, $X \in Forge_c(E, \mathcal{K})$, $(X_i = u)^\bullet$, $(X = u)^{sm}$, $(\forall j\ X_i \neq u)\ (Y \notin Forge_c(E, \mathcal{K}))$, *where* $X \in \mathcal{X}$, $Y \in \mathcal{X} \cup \mathcal{X}_\mathcal{I}$, $X_i \in \mathcal{X}_\mathcal{I}$, $u \in \mathcal{T}$, $j \in Var_\mathcal{I}(u)$, $E \subset \mathcal{T}$ *and* $\mathcal{K} \subset \mathcal{T}$

We will prove that at each step of our algorithm, the normalized constraint system contains only solved constraints.

Application to the Asokan-Ginzboorg Protocol

Consider the constraint system to normalize given in Section 10.1. We only focus on the first step of this protocol : $mpair(i, \langle L, \{E_i\}_p\rangle) \in Forge(E_1, \mathcal{K}_1)$ where $E_1 = \{mpair(t, \langle l, \{e\}_p\rangle)\}$, and $\mathcal{E} = \emptyset$.

$$
\begin{array}{rl}
 & mpair(i, \langle L, \{E_i\}_p\rangle) \in Forge(E_1, \emptyset) \\
\longrightarrow & (mpair(i, \langle L, \{E_i\}_p\rangle) \in Forge_c(E_1, \emptyset)) \\
 & \vee (mpair(i, \langle L, \{E_i\}_p\rangle) \in Sub(mpair(t, \langle l, \{e\}_p\rangle), E_1, \mathcal{E}, \emptyset)) \text{ by } Rule\ 9 \\
\longrightarrow & \forall i\ ((\langle L, \{E_i\}_p\rangle \in Forge(E_1, \emptyset)) \vee (mpair(i, \langle L, \{E_i\}_p\rangle) = mpair(t, \langle l, \{e\}_p\rangle))) \\
 & \vee (mpair(i, \langle L, \{E_i\}_p\rangle) \in Sub_d(mpair(t, \langle l, \{e\}_p\rangle), E_1, \mathcal{E}, \emptyset))) \text{ by } Rules\ 13,\ 15 \\
\longrightarrow & \forall i\ \forall j\ ((\langle L, \{E_i\}_p\rangle \in Sub(mpair(t, \langle l, \{e\}_p\rangle), E_1, \mathcal{E}, \emptyset)) \vee (\langle L, \{E_j\}_p\rangle = \langle l, \{e\}_p\rangle) \\
 & \vee (mpair(i, \langle L, \{E_i\}_p\rangle) \in Sub_d(\langle l, \{e\}_p\rangle, E_1, \mathcal{E}, \emptyset)) \vee (\langle L, \{E_i\}_p\rangle \in Forge_c(E_1, \emptyset))) \\
 & \text{by } Rules\ 9,\ 19,\ 15,\ 22,\ 24 \\
\longrightarrow & \forall i\ \forall j\ ((L \in Forge(E_1, \emptyset) \wedge \{E_i\}_p \in Forge(E_1, \emptyset)) \vee ((L = l)^{sm} \wedge E_j = e) \\
 & \vee (\langle L, \{E_i\}_p\rangle \in Sub(\langle l, \{e\}_p\rangle, E_1, \mathcal{E}, \emptyset))) \text{ by } Rules\ 10,\ 15,\ 22,\ 19,\ 16,\ 20,\ 23 \\
\longrightarrow & \forall i\ \forall j\ ((L \in Forge(E_1, \emptyset) \wedge E_i \in Forge(E_1, \emptyset) \wedge p \in Forge(E_1, \{E_i\}_p)) \\
 & \vee (L \in Forge(E_1, \emptyset) \wedge \{E_i\}_p \in Sub_d(mpair(t, \langle l, \{e\}_p\rangle), E_1, \mathcal{E}, \emptyset)) \\
 & \vee ((L = l)^{sm} \wedge E_i = e) \vee ((L = l)^{sm} \wedge E_j = e)) \text{ by } Rules\ 9,\ 11,\ 15,\ 22,\ 23,\ 16,\ 17,\ 20 \\
\longrightarrow & \forall i\ \forall j\ ((L \in Forge(E_1, \emptyset) \wedge E_i = e) \vee ((L = l)^{sm} \wedge E_i = e) \vee ((L = l)^{sm} \wedge E_j = e)) \\
 & \text{by } Rules\ 15,\ 19,\ 16,\ 17,\ 20,\ 22,\ 23 \\
\longrightarrow & \forall i\ \forall j\ ((L \in Forge(E_1, \emptyset) \wedge (E_i = e)^m) \vee ((L = l)^{sm} \wedge (E_i = e)^m) \\
 & \vee ((L = l)^{sm} \wedge (E_j = e)^m)) \text{ by } Rule\ 4
\end{array}
$$

5 Verification of Well-Tagged Protocols

We introduce here the verification algorithm and the results that state the correctness and the completeness of the inference rules of Section 4 and decidability for protocols without $mpair(,)$'s and indexed variables. Given a set R of inference rules and a formula F we say that $R(F)$ is a *closure* of F by R if it is derived by a finite number of applications of rules in R and no rule can be further applied to $R(F)$. First of all, we define a reduction of equalities chain in an environment:

Notation 2. *($\lceil\mathcal{E}\rceil$) We note $\lceil\mathcal{E}\rceil$ the closure of $\mathcal{E}$ by the following rules:*

$$\begin{array}{rcl} X = Y \wedge Y = u & \longrightarrow & X = u \wedge Y = u \\ X_i = Y_i \wedge Y_j = u & \longrightarrow & X_i = u\delta_{Q,i}^{k'} \wedge Y_j = u \end{array}$$

for $X, Y \in \mathcal{X} \cup \mathcal{X}_\mathcal{I}$, $X_i, Y_j \in \mathcal{X}_\mathcal{I}$ and $u \notin \mathcal{X} \cup \mathcal{X}_\mathcal{I}$.

Second, we introduce the normalisation function denoted by $S \mapsto (S)\downarrow$. A normalisation of a constraint system can be defined when some closures can be computed as follows using a subset of the inference rules. This normalization operates in two main phases:

Definition 25 (Normalization function)**.** *Let S be a block system. We denote by SR the whole set of inference rules except Rule 4. We assume that we can compute:*
Phase 1: *S_1, a closure of S through SR except Rules 26 and 27;*
Labelling: *S_2, a closure of S_1 through Rules 4 and 5* only*;*
Phase 2: *S_3, a closure of S_2 through SR. This closure is denoted by$(S)\downarrow$.*

The *Labelling* step adds labels for creating master constraints (Rule 4), making sure to always favour labelling of equality constraints to a forge constraints (Rule 5). While Phases 1 and 2 are similar by the rules they use, their behaviors differ: when used in our algorithm for a step $R_i \Rightarrow S_i$, Phase 1 will never use any constraint interleaving rule with a master constraint $\mathcal{M}(\overrightarrow{X})$ with $\mathcal{L}(X) = E_{i-1}$. This means that during Phase 1, the variables with maximum level cannot be replaced yet from a block to an other, simply because none of them have master constraints yet. However, the second phase do not have this limitation. The verification algorithm is the following:

Algorithm 1. *Let $P = \{R'_\imath \Rightarrow S'_\imath | \imath \in J\}$ be Well-Tagged, $Sec \in \mathcal{T}$, and $S_0 \subset \mathcal{T}_g$.*

1. *Guess a correct execution order $\pi : J \rightarrow 1..k$.*
2. *Let $R_i \triangleq R'_{\pi^{-1}(i)}$ and $S_i \triangleq S'_{\pi^{-1}(i)}$ $\forall i \in 1..k$. Let $R_{k+1} \triangleq Sec$.*
3. *Let $CBS_0 \triangleq \forall Q\, \exists R\, \top$, with $Q = R = \emptyset$, be the initial constraint system.*
4. *For i from 1 to $k+1$:*
   (a) *Assume that $CBS_{i-1} \triangleq \forall Q \exists R\, B_1 \vee B_2 \vee .. \vee B_p$;*
   (b) *Let $ctr_i \triangleq R_i \in Forge(S_0, S_1, .., S_{i-1}, \emptyset)$;*
   (c) *Let $\mathcal{E}_i = \bigcup_{\overrightarrow{X}} \mathcal{M}(CBS_{i-1}, \overrightarrow{X})$ and for all $j = 1, ..p$, $X, Y \in \mathcal{X} \cup \mathcal{X}_\mathcal{I}$, $\mathcal{E}_{i,j} = \lceil \mathcal{E}_i \cup \mathcal{SM}(B_j, \mathcal{X}) \rceil \backslash \{(X = Y)\}$;*
   (d) *Let $CBS_i \triangleq (\forall Q \exists R\, (B_1 \wedge ctr_i, \mathcal{E}_{i,1}) \vee .. \vee (B_p \wedge ctr_i, \mathcal{E}_{i,p}))\downarrow$*
5. *Test Satisfiability of CBS_{k+1} (return insecure iff satisfiable).*

Note that sets $\mathcal{E}_i, \mathcal{E}_{i,j}$ denotes respectively the set of master constraints for vector variables and the set of submaster constraints for variables of block B_j, both with variables of level strictly included in E_{i-1}. Notation $\top$ represents true. The algorithm chooses a "possible" protocol run represented by π, and tests if after this run Sec is derivable by the intruder for some length e of $mpair(,)$. We test this by increasing the initial constraint system CBS_0 with each protocol step successively, and by normalising the resulting constraint system at each step. This step-by-step normalisation is required by our inference rules which assumes that master and sub-master constraints for previous steps have been already computed.

Results obtained for our inference system are the followings:

Theorem 2. *The insecurity problem for Well-Tagged protocols with Autonomous Keys is decidable.*

The proof of Theorem 2 follows from Lemmas 3, 4 and 5.

Lemma 3. *(Correctness and Completeness of Normalization)*
Let CBS_i and ctr_i $(i = 1..k+1)$ be as in the verification algorithm, for some Well-Tagged *protocol P. Then for all e, $[\![CBS_{i-1} \wedge ctr_i]\!]^e_\emptyset = \left|\!\left[(CBS_{i-1} \wedge ctr_i)\downarrow \right]\!\right|^e_\emptyset$*

Lemma 4. *(Termination of Normalization)*
Algorithm 1 terminates for Well-Tagged protocols with Autonomous Keys.

Lemma 5. *(Satisfiability of normalized form)*
When Algorithm 1 is applied to a Well-Tagged protocol P with autonomous keys, the satisfiability of the resulting normalised constraint system can be decided.

The proof of Lemma 3 is in Section 6. The proof of Lemma 4 is in Section 8. The proof of Lemma 5 is in Section 9. It is worth also to notice that our algorithm always terminates for protocols without $mpair(,)$'s and without index variables (See Section 7), thus showing that our procedure is an extension of protocol analysis in the basic case. Also, note that the satisfiability of CBS_{k+1} will be trivial to check for protocols without $mpair(,)$. Moreover, the satisfiability of CBS_{k+1} is also trivial to check in our example, the Asokan-Ginzboorg protocol.

6 Correctness and completeness

The aim of this section is to show that the different reduction rules preserve the set of solutions. That is, we say that a rule $F_1 \rightarrow F_2$ over a constraint system is complete when $\forall e$, $[\![F_1]\!]^e_\emptyset \subseteq [\![F_2]\!]^e_\emptyset$, and correct when $\forall e$, $[\![F_1]\!]^e_\emptyset \supseteq [\![F_2]\!]^e_\emptyset$. Note also for a rule r, post(r) denotes the right hand side of r and pre(r) denotes the left hand side of r.

We first need a notion of *variable level*:

Definition 26 (Variable Level, Vector Level)**.** *Assuming that P, Sec, S_0 are as in Definition 6 and π is a correct execution order for the protocol, we denote by $E_i = S_0, S_1, .., S_i$ for any $i \in 1..k$. Let $A \in \mathcal{X} \cup \mathcal{X}_\mathcal{I}$. Then, $\mathcal{L}(A)$ is the smallest set E_i for $i = 1 \ldots k$ such that $A \leq R_{i+1}$. We extend the notion of level to variable vector in the following way: Let $\overrightarrow{X} \in \overrightarrow{\mathcal{X}}$. Then, $\mathcal{L}(\overrightarrow{X})$ is the smallest set E_i for $i = 1 \ldots k$ such that $\exists m \in \mathcal{I}$ with $E_i = \mathcal{L}(X_m)$.* □

We give some properties that are preserved by our constraint solving rules and permit us to prove the correctness and the completeness of the algorithm.

6.1 Properties of the Rules System

We show invariants that state properties satisfied by terms or constraints occuring in any constraint system derived at some step in the normalization.

Invariant 1. *$\#Var_\mathcal{I}(t) \leq 1$ for $t \in \mathcal{T}$.*

Invariant 1 will be used for the correctness and completeness proof of Rule 35.

Proof. Initially constraints are of the form $t \in Forge(E, \mathcal{K})$. According to the property of *mpair* autonomy over $\mathcal{P}$ (See Definition 8), we have $Var_\mathcal{I}(t) = \emptyset$. Thus Invariant 1 is satisfied by the initial constraints. We show that the application of any rule of SR preserves this invariant.
Rules 1, 4, 5, 6 and 8 do not change any constraint then Invariant 1 remains satisfied.
Rule 2 changes a constraint into another one using the same terms as for the left-hand side of the rule. Then, Invariant 1 is preserved. Rule 3 eliminates the whole block. Then, the invariant still holds. Rule 7 adds a new constraint $Y_j = Z$ to the block and this constraint satisfies the Invariant.
Rule 9 transforms a constraint $t \in Forge(E, \mathcal{K})$ to either a $Forge$ constraint $t \in Forge_c(E, \mathcal{K})$ conserving the same t or to a Sub constraint using the two terms t and $w \in E$. By induction hypothesis we have $\#Var_\mathcal{I}(t) \leq 1$. Besides, according to the property of *mpair* autonomy over $\mathcal{P}$, we have $Var_\mathcal{I}(w) = \emptyset$ and then $\#Var_\mathcal{I}(w) = 0$. Therefore, the invariant is preserved.
Rule 10 decomposes the term $\langle t_1, \ldots, t_m \rangle$ to be forged into its subterms t_i, $i = 1 \ldots m$.
Since $\#Var_\mathcal{I}(\langle t_1, \ldots, t_m \rangle) \leq 1$ (by induction hypothesis the invariant is satisfied on the block system to be rewritten), we have $\#Var_\mathcal{I}(t_i) \leq 1$ $\forall i = 1 \ldots m$.

A similar reasoning applies to Rules 11 and 12.
For Rule 13, according to the property of *mpair* autonomy over terms we have $Var_{\mathcal{I}}(t) = \{k\}$. Thus the invariant is satisfied.
Rule 14 eliminates the whole block. Then, the invariant still holds. Rule 15 transforms a Sub constraint either to another Sub constraint or $Equality$ one while preserving the two terms t and u. The invariant is then preserved in the two cases.
In Rule 16, the Sub constraint is transformed into a Sub constraint with the same term t and subterm t_i of $\langle t_1, \ldots, t_m \rangle$. Since $\#Var_{\mathcal{I}}(\langle t_1, \ldots, t_m \rangle) \leq 1$ (by induction hypothesis), we have $\#Var_{\mathcal{I}}(t_i) \leq 1 \; \forall i = 1 \ldots m$.
A similar reasoning applies to Rules 17 and 18 (with addition of $Forge$ constraint with a subterm of the initial term t).
For Rule 19, according to the property of *mpair* autonomy over terms we have $Var_{\mathcal{I}}(u) = \{k\}$. Thus, the invariant is satisfied. Rule 20 eliminates the whole block. Then, the Invariant remains valid. Rules 21 and 22 eliminate either the constraint ($\top$) or the block ($\bot$). Then, the invariant is preserved.
Rule 23 transforms an Equality constraint with two terms t and t' into equality constraints with subterms of t and t'. Then, the invariant is preserved.
For Rule 24, according to the property of *mpair* autonomy over terms we have $Var_{\mathcal{I}}(u) = \{k\}$ and $Var_{\mathcal{I}}(v) = \{l\}$. Thus, the invariant is satisfied.
Rule 25 replaces an index variable by another index variable in a block. Thus, the invariant remains satisfied.
In Rules 26, 27, 28, 29, 30 and 31, the block system resulting from the rewriting has the same terms as the one that is reduced. Thus the invariant is satisfied.
Rule 32 eliminates the block and the invariant obviously holds.
Rules 33, 34 and 35, transforms a block system into one built with the same terms as the initial one, reduced modulo some index replacement. Thus the invariant is satisfied. □

Invariant 2. *We say that a constraint ctr is a constraint for X (or X has a constraint ctr) if $ctr = (X \in Forge_c(E, \mathcal{K}))$ or $ctr = (X = u)$.*
$\forall X \in \mathcal{X} \cup \mathcal{X}_{\mathcal{I}}$, for any rule r in $SR \cup \{Rule\ 4\}$, if X has a constraint in $pre(r)$, then it has a constraint in any B with $B \in post(r)$

Invariant 2 is used in the proof of Corollaries 1 and 2.

Proof. We show that the application of any rule of $SR \cup \{Rule\ 4\}$ preserves Invariant 2.
Rule 2 transforms a constraint for Y into another constraint for Y without elliminating the other constraints. Thus, we still have constraints for both X and Y. Rule 3 eliminates the block. Then, the invariant holds. Rules 4, 5, and 8 do not change constraints. Rule 6 transforms a constraint for X into another constraint for X without elliminating other constraints. Thus, we still have constraints for both X and Y_j. Rule 7 transforms a constraint for X into another constraint for X, while adding a new constraint for Y_j. We conclude that control rules satisfies the invariant.
Rule 9 may add a constraint for a variable but can not eliminate ones. Thus, the invariant holds. The other rules of Group G_2 do not treat constraints for variables. Then, the invariant remains valid. Rule 15 may add a constraint for a variable but can not eliminate ones. Then, the invariant is satisfied. The other rules of Group G_3 do not treat constraints for variables. Thus, Group G_3 preserves the invariant. Rule 23 may add a constraint for a variable but can not eliminate ones. Thus, the invariant holds. The other rules of Group G_4 do not manage constraints for variables. Thus, Group G_4 preserves the invariant.
Rules 26, 27, 30 and 31, do not eliminate constraints for variables. Rule 28 eliminates a constraint for X_i ($X_i \in Forge_c(E', \mathcal{K})$) but it remains another constraint for X_i : $X_i = u$. We reason similarly for Rule 29. Rule 32 eliminates the block. We conclude that interleaving rules inside a block preserve Invariant 2.
In Rule 34, the constraint for X_m: $X_m \in Forge_c(E', \mathcal{K})$ would be transformed in either $X_m \in Forge_c(E', \mathcal{K})$ or $X_m = u_0\delta_0$. Then, in both cases, we still have a constraint for X_m. In Rule 35, the constraint for X_m: $X_m = v$ is either preserved or transformed into another constraint : $X_m = u_0\delta_0$. Then, in both cases, we still have a constraint for X_m. We conclude that interleaving rules between different blocks preserve Invariant 2, and therefore it is preserved by $SR \cup \{Rule\ 4\}$. □

Invariant 3. *For a constraint $(t \in Sub_{\bullet}(w, E, \mathcal{E}, \mathcal{K}))$, we have $\forall X \leq w$ where $X \in \mathcal{X} \cup \mathcal{X}_{\mathcal{I}}$, $\mathcal{L}(X) \subset E$.*

Invariant 3 is used in the proof of Proposition 21.

Proof. Initially, Sub constraints are obtained from $Forge$ ones by Rule 9. We get a constraint $t \in Sub(w, E, \mathcal{E}, \mathcal{K})$ where $w \in E$. Then, $\forall X \leq w$ and according to the notion of correct execution, and the definition of $\mathcal{L}(X)$,

we have $\mathcal{L}(X) \subset E$. We only focus on rules treating Sub constraints. Rule 15 transforms a constraint $t \in Sub(u, E, \mathcal{E}, \mathcal{K})$ into a constraint $t \in Sub_d(u, E, \mathcal{E}, \mathcal{K})$ without modifying neither u nor E. Then, Invariant 3 remains satisfied. In Rule 16, since the invariant is satisfied for $\langle t_1, \ldots, t_m \rangle$ and E, then it remains satisfied for a subterm of $\langle t_1, \ldots, t_m \rangle$: t_i and E. The same reasoning is valid for Rules 17, 18 and 19. Rules 20 and 32 eliminate the whole block, then, the invariant remains valid. Rule 32 eliminates the whole block. Then, Invariant 3 is satisfied. In Rule 31, we get a Sub constraint $t \in Sub(w, E', \mathcal{E}, \mathcal{K})$ where w is given by the constraint $X = w$ which belongs to $\mathcal{E}$. However, by construction of $\mathcal{E}$, $\mathcal{L}(w) \subset E'$. Finally, for Rule 33, we reason similarly to Rule 31. □

In what follows, properties and lemmas introduced are limited to the first phase of the computation of a certain CBS_i.

Invariant 4. *$\forall ctr$ in the computation of some CBS_i at phase 1, $\forall X \in \mathcal{X} \cup \mathcal{X}_\mathcal{I}$ s.t $\mathcal{L}(X) = E_{i-1}$, if $ctr = t \in Sub_.(w, E, \mathcal{E}, \mathcal{K})$ s.t $X \leq t$ and $\forall Y \leq w$ then $\mathcal{L}(Y) \subset \mathcal{L}(X)$.*

Invariant 4 is used in the proof of Proposition 1.

Proof. Initially, constraints are $Forge$ ones. Then, the invariant is satisfied. We prove that Invariant 4 is preserved at each application of a rule of our inference rules. The first group does not manage Sub constraints. Then, the invariant remains satisfied. Rule 9 manages a constraint $t \in Sub(w, E, \mathcal{E}, \mathcal{K})$ where $w \in E$. However, according to the notion of correct execution, and by definition of $\mathcal{L}(Y)$, $E \subset E_i$ and $\mathcal{L}(Y) \subseteq E$. Thus, $\mathcal{L}(Y) \subset \mathcal{L}(X)$, which satisfies the invariant. The other rules of Group G_2 do not manage Sub constraints. Therefore, Group G_2 satisfies the invariant. Rules 15 transforms a Sub constraint into another Sub_d constraint with the same t and u. Thus, the invariant still holds. Rule 20 eliminates the whole block. Then, the invariant remains valid. The other rules of Group G_3 decompose the term inside the Sub constraint. Then, the invariant still holds. Thus, Group G_3 satisfies the invariant. Group G_4 does not treat Sub constraints. Then, the invariant still holds. Rule 31 generates a constraint $t \in Sub_d(w, E, \mathcal{E}, \mathcal{K})$. However, w comes from a sub-master constraint $(X = w) \in \mathcal{E}$. By construction of the environment, $\mathcal{L}(Y) \subset \mathcal{L}(X)$, which satisfies the invariant. The other rules of Group G_5 do not manage Sub constraints. Then, Group G_5 satisfies the invariant. Rule 33 generates a constraint $t \in Sub_d(u\delta, E', \mathcal{E}, \mathcal{K})$. However, $u\delta$ comes from a master constraint $X_i = u \in \mathcal{E}$. By construction of $\mathcal{E}$, $\mathcal{L}(Y) \subset \mathcal{L}(X)$, which satisfies the invariant. The other rules of Group G_6 do not treat Sub constraints. Thus, Group G_6 satisfies the invariant. We conclude that Invariant 4 is satisfied by our inference rules. □

Definition 27. *We say that a constraint ctr' has type* (1), (1′),(2),(2′) *or* (3) *for a variable X if*

$$\begin{array}{lll} ctr' = (t \in Forge(E, \mathcal{K})) & \text{where } X \leq t, & (1) \\ \text{or} \quad ctr' = (t \in Forge_c(E, \mathcal{K})) & \text{where } X \leq t, & (1') \\ \text{or} \quad ctr' = (t \in Sub(u, E, \mathcal{E}, \mathcal{K})) & \text{where } X \leq t & (2) \\ \text{or} \quad ctr' = (t \in Sub_d(u, E, \mathcal{E}, \mathcal{K})) & \text{where } X \leq t & (2') \\ \text{or} \quad ctr' = (u = v) & \text{where } X \leq u \text{ or } X \leq v & (3) \end{array}$$

Invariant 5. *For any rule r in SR except Rules 26 and 27 (ie. Rules of the phase 1), $\forall X \in \mathcal{X} \cup \mathcal{X}_\mathcal{I}$ s.t $\mathcal{L}(X) = E_{i-1}$, if pre(r) contains a constraint of type* (1), (1′), (2), (2′) *or* (3) *for X then $\forall B \in post(r)$, B contains a constraint of type* (1), (1′), (2), (2′) *or* (3) *for X.*

Proof. Rules 3, 14, 20,22, 32 eliminate the whole block. Then the invariant is valid. Rules 1, 2, 4, 5 and 8 do not change constraints. Then, the invariant remains valid.
In Rule 6 and 7, constraints for other variables than Y_j do not change. For Y_j, there exists a constraint of type (3). Thus, the invariant still holds.
For Rule 9, we obtain either a constraint of type (1′) or a constraint pf type (2) for X. For Rules 10, 11, 12 and 13, the term t is decomposed. We obtain as a result, a constraint of type (1) for X. We conclude that $Forge$ constraints validate the invariant.
For Rule 15, we obtain either a constraint of type (3) or a constraint of type (2′) for X. For Rules 16, 17, 18 and 19, the term t is decomposed. We obtain as a result, a constraint of type (2) for X. We conclude that Sub constraints validate the invariant.
In Rules 21 and 25, pre(r) does not contain a constraint of type (1), (1′),(2),(2′) or (3) for X. For Rules 23 and 24, the term t is decomposed. We obtain as a result, a constraint of type (3) for X. We conclude that $Equality$ constraints validate the invariant.
For Rule 28, $(X_i \in Forge_c(E, \mathcal{K}))$ is tranformed to $(u \in Forge_c(E, \mathcal{K}))$ but it still exists a constraint of type

(3) for X_i which is $(X_i = u)^\bullet$. The same reason is valid for Rule 29. In Rule 30, there is a constraint of type $(1')$ for A. We conclude that rules for interleaving in the same block preserve the invariant.
Rules 33 and 34 preserve the same constraint given in $\text{pre}(r)$. Rule 35 uses the same terms in equality constraints. Then, it transforms a constraint of type (3) for variables in v or X_m into other constraints of the same type. We conclude that rules for interleaving in different blocks preserve the invariant. □

Proposition 1. *At the end of Phase 1 in the computation of CBS_i, for all $X \in \mathcal{X} \cup \mathcal{X}_\mathcal{I}$, for all blocks B there is a constraint $ctr \in B$ such that $ctr = (X \in Forge_c(E, \mathcal{K}))$ or $ctr = (X = u)$.*

Proof. We first introduce the following claim:

Claim 1. *$\forall ctr$ in the computation of some CBS_i at phase 1, $\forall X \in \mathcal{X} \cup \mathcal{X}_\mathcal{I}$ s.t $\mathcal{L}(X) = E_{i-1}$, if $ctr = (Y = u)$ and $X \leq u$ then $\mathcal{L}(Y) < \mathcal{L}(X)$.*

Proof. We show by contradiction that if $ctr = (Y = u)$ and $X \leq u$ then $\mathcal{L}(Y) < \mathcal{L}(X)$. Let $ctr = (Y = u)$ such that $\mathcal{L}(Y) = \mathcal{L}(X)$. Consider a derivation $d = d'_j.L_j.d_j$ such that $(Y = u) \in \text{post}(L_j)$. Then, $\exists l < j$ such that $(u' = v') \in \text{post}(L_l)$ where $Y \leq v'$ and $u \leq u'$ and $\text{pre}(L_l) \neq (u" = v")$ where $u' < u"$ and $v' < v"$. $(u' = v')$ is obtained either by transforming a *Sub* constraint to an *Equality* one by Rule 15, or by interleavings (Rule 35). Note that Rules 33 and 34 generates equality contraints bu not suitable for our case since they may not contain X since they are in $\mathcal{E}$ and $\mathcal{L}(X) = E_{i-1}$.
1. Case $\text{pre}(L_l) = (u' \in Sub(v', E, \mathcal{E}, \mathcal{K}))$.
Since $X < u'$, $\mathcal{L}(X) = E_{i-1}$ and we compute CBS_i, according to Invariant 4, we have $\mathcal{L}(Y) \subset \mathcal{L}(X)$ which contradicts the hypothesis: $\mathcal{L}(Y) = \mathcal{L}(X)$. The same reasoning is valid if $\text{pre}(L_l) = (v' \in Sub(u', E, \mathcal{E}, \mathcal{K}))$.
2. case $L_l = R$ 35, then $\text{pre}(L_l) = (X_m = u')$ and as master constraint we have $(X_i = v')$. Since we are in the first phase, that is before labeling master or submaster constraints for variables of level E_{i-1}, then $\mathcal{L}(v') \subset E_{i-1}$. Since $\mathcal{L}(X) = E_{i-1}$ and $Y < v'$ then $\mathcal{L}(Y) \subset \mathcal{L}(X)$ which is in contradiction with our hypothesis $\mathcal{L}(Y) = \mathcal{L}(X)$. □

Note that at the end of Phase 1 in the computation of CBS_i, we only have solved constraints (Hypothesis H_1). We show by contradiction that there exists a constraint ctr for X in each block of the constraint system S. Let B be a block of S such that $\nexists ctr \in B$ for X (Hypothesis H_2). According to Invariant 5, $\exists ctr' \in B$ such that ctr' has type (1), $(1')$, (2), $(2')$ or (3) for X. There are five cases. In the first one, $ctr' = (t \in Forge(E, \mathcal{K}))$ where $X < t$, otherwise (when $t \in \mathcal{X} \cup \mathcal{X}_\mathcal{I}$) $ctr = ctr'$ which is in contradiction with hypothesis H_1. Then, Rule 9 may be applied which is in contradiction with hypothesis H_1.
In the second case, $ctr' = (t \in Forge_c(E, \mathcal{K}))$ where $X < t$. Then, Rules 10, 11, 12 and 13 may be applied which is in contradiction with hypothesis H_1.
In the third case, $ctr' = (t \in Sub(u, E, \mathcal{E}, \mathcal{K}))$ where $X \leq t$. Then, Rule 15 may be applied which contradicts hypothesis H_1. In the fourth case, $ctr' = (t \in Sub_d(u, E, \mathcal{E}, \mathcal{K}))$ where $X \leq t$. There are two cases. In the first one, $u \notin \mathcal{X} \cup \mathcal{X}_\mathcal{I}$. Then, Rules 16, 17, 18, 19 and 20 may be applied what contradicts hypothesis H_1. In the second case, $u = Y \in \mathcal{X} \cup \mathcal{X}_\mathcal{I}$. Then, according to Invariant 4, we have $\mathcal{L}(Y) \subset \mathcal{L}(X)$. Thus, $\exists ctr_3 \in \mathcal{E}$ such that $ctr_3 = (Y \in Forge(E3, \mathcal{K}'))$ or $ctr_3 = (Y = u3)$ (by construction of $\mathcal{E}$). Therefore, Rules 31, 32 and 33 may be applied which contradicts hypothesis H_1. In the fifth case, $ctr' = (u = v)$ where $X \leq v$ or $X \leq u$. There are two cases. In the first one, $u \notin \mathcal{X} \cup \mathcal{X}_\mathcal{I}$. Then, Rules 22, 23 and 24 may be applied what contradicts hypothesis H_1. In the second case, $u \in \mathcal{X} \cup \mathcal{X}_\mathcal{I}$. If $u = X$ then $ctr' = ctr$ which contradicts hypothesis H_2. If $u = Y$ and $X \leq v$ then according to Claim 1, $\mathcal{L}(Y) \subset \mathcal{L}(X)$. Then, $\exists ctr" \in \mathcal{E}$ for Y and therefore Rules 27 and 35 may be applied which contradicts hypothesis H_1. □

Corollary 1. *At the end of every phase, for every block B, $\forall X_i \in \mathcal{X}_\mathcal{I}$, there exists a single master constraint $(ctr)^m \in B$ such that $ctr = (X_i \in Forge_c(E, \mathcal{K}))$ or $ctr = (X_i = u)$.*

Proof. For Phase 1 this is an immediate consequence of Proposition 1. The unicity of the master constraint is guaranteed by the condition $B \cap \mathcal{M}(\overrightarrow{X}) = \emptyset$ of Rule 4. At the begining of Phase 2, for a block B, $\forall X_i \in \mathcal{X}_\mathcal{I}$, there exists a single constraint $(ctr)^m \in B$ such that $ctr = (X_i \in Forge_c(E, \mathcal{K}))$ or $ctr = (X_i = u)$ since Invariant 1 is preserved by Phase 1.
According to Invariant 2, our system of rules preserve constraints for X_i (*Forge* or *Equality* constraints for X_i) which are potential master constraints for X_i. Our rules never eliminate a master constraint for a variable. They can change a master constraint for a variable after introducing new candidate constraints (*Equality* constraints for the variable). This is managed by Rule 5. Rule 5 labels a new master constraint for $\overrightarrow{X}$

$(X_j = u)$ while eliminating the label of the old master constraint. Thus, in that case we still have a single master constraint for $\overrightarrow{X}$. □

Corollary 2. *At the end of every phase, for every block B, $\forall X \in \mathcal{X}$, there exists a constraint $ctr \in B$ such that $ctr = (X \in Forge_c(E, \mathcal{K}))$ or $ctr = (X = u)$.*

Proof. The proof follows directly from Proposition 1 and Invariant 2. □

6.2 Correctness and Completeness of Rules

We prove correctness and completeness of each rule of our system for an index substitution τ and a value e for n. Correctness and completeness of Rules 1, 6, 7, 21 and 22 is trivial. For Rules 4, 5 and 8, since the semantics of a constraint is the same with or without the label, and since these rules only modify labels then they are complete and correct.

Proposition 2. *Rule 2 is correct and complete.*

Proof.

$$\sigma \in [\![(X = u)^{sm} \wedge (Y = X)]\!]^e_\tau \quad \text{iff} \quad \overline{X}^e\tau\sigma = \overline{u}^e\tau\sigma \wedge \overline{Y}^e\tau\sigma = \overline{X}^e\tau\sigma \quad \text{iff}$$
$$\overline{X}^e\tau\sigma = \overline{u}^e\tau\sigma \wedge \overline{Y}^e\tau\sigma = \overline{u}^e\tau\sigma \quad \text{iff} \quad \sigma \in [\![(X = u)^{sm} \wedge (Y = u)]\!]^e_\tau$$

□

Proposition 3. *Rule 3 is correct and complete.*

Proof. Let B be a block with $(X = u) \in B$, $Y < u$, and $Y \sqsubset X$ in B. Then, $\forall\tau$, $\forall e$, $\forall\sigma \in [\![B]\!]^e_\tau$, we have $\overline{Y}^e\tau\sigma < \overline{X}^e\tau\sigma$ thanks to $Y \sqsubset X$ and $\overline{Y}^e\tau\sigma < \overline{X}^e\tau\sigma$ thanks to $Y < u$ and $(X = u) \in B$. This is impossible, and therefore, $[\![B]\!]^e_\tau = [\![\bot]\!]^e_\tau$ □

Correctness and completeness of Rule 9 follows from Propositions 4 and 5 below.

Proposition 4.
$[\![B \wedge t \in Forge(E, \mathcal{K})]\!]^e_\tau \subseteq \left[\!\!\left[(B \wedge t \in Forge_c(E, \mathcal{K})) \vee \bigvee_{w \in E} (B \wedge t \in Sub(w, E, \mathcal{E}, \mathcal{K})) \right]\!\!\right]^e_\tau$.

Proof. Let $\sigma \in [\![B \wedge t \in Forge(E, \mathcal{K})]\!]^e_\tau$. Then, $\sigma \in [\![B]\!]^e_\tau \cap [\![t \in Forge(E, \mathcal{K})]\!]^e_\tau$. But, $[\![t \in Forge(E, \mathcal{K})]\!]^e_\tau = [\![t \in Forge_c(E, \mathcal{K})]\!]^e_\tau \cup ([\![t \in Forge(E, \mathcal{K})]\!]^e_\tau \backslash [\![t \in Forge_c(E, \mathcal{K})]\!]^e_\tau)$. Thus, we will prove that $\exists w \in E$ s.t. $\sigma \in [\![t \in Sub(w, E, \mathcal{E}, \mathcal{K})]\!]^e_\tau$ if $\sigma \in ([\![t \in Forge(E, \mathcal{K})]\!]^e_\tau \backslash [\![t \in Forge_c(E, \mathcal{K})]\!]^e_\tau)$. We first prove some lemmas. We will prove this proposition by using the following lemma that follows from Proposition 2 from [RT03].

Lemma 6. *Let $t \in Dy(E, \mathcal{K})$ and $\gamma \in Dy_c(E, \mathcal{K})$ be given with $D_\gamma(E) \in NRD$. Then, there is a derivation $D'_t(E) \in NRD$ verifying $L_d(\gamma) \notin D'$.*

Proof. The proof is exactly the same as mentioned in [RT03] with the difference that $D_\gamma(E)$ is not a minimal derivation but rather a non-redundant derivation. The proof remains valid as we have no useless rules in $D_\gamma(E)$ since it is a non-redundant derivation. Moreover, $D'_t(E)$ as it is constructed in [RT03] is a general derivation. However, according to Remark 1, there exists an equivalent non-redundant derivation for $D'_t(E)$. □

In our context, Lemma 6 says that for any term t, a set of terms E, a set of terms $\mathcal{K}$ and a variable $X \in \mathcal{X} \cup \mathcal{X}_\mathcal{I}$, and for any e, τ and σ, if $\overline{t}^e\tau\sigma \in Dy(E\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$ and $X\tau\sigma \in Dy_c(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$ then there exists a derivation D' building $\overline{t}^e\tau\sigma$ from $\overline{E}^e\tau\sigma$ where $X\tau\sigma$ is never decomposed. This can be easily generalized from singleton $\{X\}$ to any set of variables instead of X.

Lemma 7. *If $\overline{t}^e\tau\sigma \in Dy_d(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$, then there exists $w \in E$ s.t. $\overline{t}^e\tau\sigma \leqslant^L_{F\sigma} \overline{w}^e\tau\sigma$ with $F\sigma \subseteq Dy(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$, $\overline{\mathcal{K}}^e\tau\sigma \cap F\sigma = \emptyset$ and $\forall X$, if $X\tau\sigma \in L$ then $X\tau\sigma \notin Dy_c(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$.*

Proof. Assume that $\overline{t}^e\tau\sigma \in Dy_d(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$. Then thanks to the Lemma 6 iterated as described above, there exists a derivation D from $\overline{E}^e\tau\sigma$ to $\overline{t}^e\tau\sigma$ such that $\forall X$, if $L_d(X\tau\sigma) \in D$ then $X\tau\sigma \notin Dy_c(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$. Moreover, $\overline{t}^e\tau\sigma \notin Dy_c(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$ by definition of $\overline{t}^e\tau\sigma \in Dy_d(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$, and thus, that D ends with a decomposition rule. Now, by iteration on the length of D starting from this last decomposition, and following the same idea as Lemma 2 in [RT03], we see that there exists $w' \in E\tau\sigma$ such that $\overline{t}^e\tau\sigma \leqslant^L_{F\sigma} w'$ with $F\sigma \subseteq Dy(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$, $\overline{\mathcal{K}}^e\tau\sigma \cap F\sigma = \emptyset$ and $\forall X$, if $X\tau\sigma \in L$ then $X\tau\sigma \notin Dy_c(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$. This is the term in $E\tau\sigma$ which is decomposed down to $\overline{t}^e\tau\sigma$ without using any term of $\overline{\mathcal{K}}^e\tau\sigma$ as a key for decryption. Finally, there exists $w \in E$ s.t. $w\tau\sigma = w'$ necessarily, and the lemma follows. □

Now we can finish the proof of Proposition 4. Since $\overline{t}^e\tau\sigma \in Dy_d(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$ thanks to Lemma 7, there exists $w \in E$ such that $\overline{t}^e\tau\sigma \leqslant^L_{F\sigma} \overline{w}^e\tau\sigma$ with $F\sigma \subseteq Dy(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$, $\overline{\mathcal{K}}^e\tau\sigma \cap F\sigma = \emptyset$ and $\forall X$, if $X\tau\sigma \in L$ then $X\tau\sigma \notin Dy_c(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$. We prove that If $\overline{t}^e\tau\sigma \leqslant^L_{F\sigma} \overline{w}^e\tau\sigma$ without any $X\tau\sigma \in L$ such that $X\tau\sigma \in Dy_c(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$ and with $F\sigma \subseteq Dy(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$ and $\overline{\mathcal{K}}^e\tau\sigma \cap F\sigma = \emptyset$, then $\sigma \in [\![t \in Sub(w, E, \mathcal{E}, \mathcal{K})]\!]^e_\tau$ by recurrence over (l, d) with l the length of $\overline{t}^e\tau\sigma \leqslant^L_{F\sigma} \overline{w}^e\tau\sigma$ and $d = 1$ if $w \in \mathcal{X} \cup \mathcal{X}_\mathcal{I}$, otherwise 0.

Base case: Assume that $l = 0$ and an arbitrary d. Then $\overline{t}^e\tau\sigma = \overline{w}^e\tau\sigma$, and thus $\sigma \in [\![t \in Sub(w, E, \mathcal{E}, \mathcal{K})]\!]^e_\tau$ according to the semantics of Sub.

Induction step: Assume that the formula above is true for any instance strictly smaller than (l, d), with $l \geq 1$. We have two cases:

-Either $w \notin \mathcal{X} \cup \mathcal{X}_\mathcal{I}$, and thus there exists a direct subterm w' of w, and there exist G, H, L_1, L_2 such that $\overline{t}^e\tau\sigma \leqslant^{L_1}_G \overline{w'}^e\tau\sigma \leqslant^{L_2}_H \overline{w}^e\tau\sigma$, with $G \cup H \subseteq Dy(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$ and the length of $\overline{t}^e\tau\sigma \leqslant^{L_1}_G \overline{w'}^e\tau\sigma$ strictly smaller than l. Therefore, $\sigma \in [\![t \in Sub(w', E, \mathcal{E}, \mathcal{K})]\!]^e_\tau$, and thus $\sigma \in [\![t \in Sub(w, E, \mathcal{E}, \mathcal{K})]\!]^e_\tau$ according to the semantics of Sub.

-Or $w = X$ for some $X \in \mathcal{X} \cup \mathcal{X}_\mathcal{I}$, and thus $X\tau\sigma \notin Dy_c(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$ since $l \geq 1$ and therefore $X\tau\sigma \in L$. Indeed, according to corollaries 1 and 2 and by construction of $\mathcal{E}$ we know that either $\exists v \notin \mathcal{X} \cup \mathcal{X}_\mathcal{I}$ s.t. $(X = v) \in \mathcal{E}$, and $X\tau\sigma = \overline{v}^e\delta\tau\sigma$ or $X\tau\sigma \in Dy_c(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$. However, the latter is impossible since we already have $X\tau\sigma \notin Dy_c(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$. Therefore, we have $\overline{t}^e\tau\sigma \leqslant^L_{F\sigma} \overline{v}^e\tau\sigma$ since $\overline{w}^e\tau\sigma = X\tau\sigma = \overline{v}^e\tau\sigma$. It follows by induction that $\sigma \in [\![t \in Sub(v, E, \mathcal{E}, \mathcal{K})]\!]^e_\tau$ since $v \notin \mathcal{X} \cup \mathcal{X}_\mathcal{I}$ and $(l, 0) < (l, 1)$. Thus, $\sigma \in [\![t \in Sub(w, E, \mathcal{E}, \mathcal{K})]\!]^e_\tau$ thanks to the second case in the definition of Sub. □

Proposition 5.
$\big[\!\big[(B \wedge t \in Forge_c(E, \mathcal{K})) \vee \bigvee_{w \in E} (B \wedge t \in Sub(w, E, \mathcal{E}, \mathcal{K})) \big]\!\big]^e_\tau \subseteq [\![B \wedge t \in Forge(E, \mathcal{K})]\!]^e_\tau$.

Proof. In order to prove Proposition 5, we need to prove Lemma 8. To do this, we first prove Proposition 6:

Proposition 6. *If $\sigma \in [\![t \in Sub(X, E, \mathcal{E}, \mathcal{K})]\!]^e_\tau$, then, either $\overline{t}^e\tau\sigma = X\tau\sigma$ or $\exists v, \delta, k, \tau'$ such that $k \notin \{t, X, \mathcal{E}\}$, $X\tau\sigma = \overline{v}^e\delta\tau'\sigma$, $v\delta \notin \mathcal{X} \cup \mathcal{X}_\mathcal{I}$ and $\sigma \in [\![t \in Sub(v\delta, E, \mathcal{E}, \mathcal{K})]\!]^e_{\tau'}$.*

Proof. Since $\sigma \in [\![t \in Sub(X, E, \mathcal{E}, \mathcal{K})]\!]^e_\tau$, then, $\exists u\ \exists F, L$ s.t $u \leqslant^L_F \overline{X}^e\tau, \overline{\mathcal{K}}^e\tau\sigma \cap F\sigma = \emptyset, F\sigma \subseteq Dy(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$. However, since $X \in \mathcal{X} \cup \mathcal{X}_\mathcal{I}$, then, $u = \overline{X}^e\tau$. Besides, we have the two cases of the semantics of Sub. In the first one, $u\sigma = \overline{t}^e\tau\sigma$. Then, $\overline{X}^e\tau\sigma = \overline{t}^e\tau\sigma$ and therefore Proposition 6 holds. In the second case, $\exists v, \delta, k, \tau'$ such that $k \notin \{t, X, \mathcal{E}\}$, $X\tau\sigma = \overline{v}^e\delta\tau'\sigma$ and $\sigma \in [\![t \in Sub(v\delta, E, \mathcal{E}, \mathcal{K})]\!]^e_{\tau'}$. However, by construction of $\mathcal{E}$, $v \notin \mathcal{X} \cup \mathcal{X}_\mathcal{I}$ which concludes the proof. □

Lemma 8. *If $\sigma \in [\![t \in Sub(w, E, \mathcal{E}, \mathcal{K})]\!]^e_\tau$, then, $\overline{t}^e\tau\sigma \in Dy_d(\overline{E}^e\tau\sigma \cup \{\overline{w}^e\tau\sigma\}, \overline{\mathcal{K}}^e\tau\sigma)$.*

Proof. We proceed by recurrence over $|\overline{w}^e\tau\sigma|$.

Base case: $|\overline{w}^e\tau\sigma| = 1$. Then, either $w \in \mathcal{C} \cup \mathcal{C}_\mathcal{I}$ or $w \in \mathcal{X} \cup \mathcal{X}_\mathcal{I}$. If $w \in \mathcal{C} \cup \mathcal{C}_\mathcal{I}$, then, $\overline{w}^e\tau\sigma = \overline{t}^e\tau\sigma$. Thus, $\overline{t}^e\tau\sigma \in Dy_d(\overline{E}^e\tau\sigma \cup \{\overline{w}^e\tau\sigma\}, \overline{\mathcal{K}}^e\tau\sigma)$. If $w = X \in \mathcal{X} \cup \mathcal{X}_\mathcal{I}$, and since $\sigma \in [\![t \in Sub(w, E, \mathcal{E}, \mathcal{K})]\!]^e_\tau$, then, according to Proposition 6, we have either (1) or (2) where:

(1) $\exists v, \delta, k, \tau'$ s.t $k \notin \{t, X, \mathcal{E}\}, X\tau\sigma = \overline{v}^e\delta\tau'\sigma, v\delta \notin \mathcal{X} \cup \mathcal{X}_\mathcal{I}$ and $\sigma \in [\![t \in Sub(v\delta, E, \mathcal{E}, \mathcal{K})]\!]^e_{\tau'}$
(2) $\overline{t}^e\tau\sigma = X\tau\sigma$

If (1) then $v\delta \in \mathcal{C} \cup \mathcal{C}_\mathcal{I}$ since $|\overline{v}^e\delta\tau'\sigma| = |\overline{w}^e\tau\sigma| = 1$. However, $\overline{v}^e\delta\tau'\sigma = \overline{t}^e\tau'\sigma$ and $\overline{t}^e\tau'\sigma = \overline{t}^e\tau\sigma$ since $k, i \notin Var_\mathcal{I}(t)$ and $\mathcal{D}om(\tau') = \mathcal{D}om(\tau) \cup \{k, i\}$ (by definition of the semantics of Sub). Therefore, $\overline{t}^e\tau\sigma \in Dy_d(\overline{E}^e\tau\sigma \cup \{\overline{w}^e\tau\sigma\}, \overline{\mathcal{K}}^e\tau\sigma)$ since $\overline{v}^e\delta\tau\sigma = \overline{t}^e\tau\sigma = \overline{w}^e\tau\sigma$.
If (2) then since $w = X$, we have $\overline{t}^e\tau\sigma \in Dy_d(\overline{E}^e\tau\sigma \cup \{\overline{w}^e\tau\sigma\}, \overline{\mathcal{K}}^e\tau\sigma)$.

Induction step: Assume Lemma 8 is true for term of size strictly smaller than $|\overline{w}^e\tau\sigma|$.
Since $\sigma \in [\![t \in Sub(w, E, \mathcal{E}, \mathcal{K})]\!]^e_\tau$, we have $\exists u\ \exists F, L$ such that $u \leqslant^L_F \overline{w}^e\tau, \overline{\mathcal{K}}^e\tau\sigma \cap F\sigma = \emptyset, F\sigma \subseteq Dy(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$ and u,F validate one of the two cases in the definition of Sub constraint solutions:

In the first case, $u\sigma = \overline{t}^e\tau\sigma$ and $u\sigma \leqslant^{L\sigma}_{F\sigma} \overline{w}^e\tau\sigma$ since $u \leqslant^L_F \overline{w}^e\tau$ (by iteration over u following the definition of $\leqslant\_$). Moreover, we have $F\sigma \subseteq Dy(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$, which leads to $\overline{t}^e\tau\sigma \in Dy_d(\overline{E}^e\tau\sigma \cup \{\overline{w}^e\tau\sigma\}, \overline{\mathcal{K}}^e\tau\sigma)$.

In the second case, $\exists v, \delta, k, \tau'$ s.t $k \notin Var_\mathcal{I}(t, w, \mathcal{E})$, $\sigma \in [\![t \in Sub(v\delta, E, \mathcal{E}, \mathcal{K})]\!]^e_{\tau'}$ and $u\sigma = \overline{v}^e\delta\tau'\sigma$. We can suppose that $v\delta \notin \mathcal{X} \cup \mathcal{X}_\mathcal{I}$. Otherwise, by Proposition 6, there exists another choice for v, δ, k, τ' such that either $v\delta \notin \mathcal{X} \cup \mathcal{X}_\mathcal{I}$ or $\overline{t}^e\tau\sigma = v\delta\tau\sigma$. The last case leads to $\overline{t}^e\tau\sigma \in Dy_d(\overline{E}^e\tau\sigma \cup \{\overline{w}^e\tau\sigma\}, \overline{\mathcal{K}}^e\tau\sigma)$ since $F\sigma \subseteq Dy(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$ and $\overline{t}^e\tau\sigma \leqslant^{L\sigma}_{F\sigma} \overline{w}^e\tau\sigma$. Thus, assuming that $v\delta \notin \mathcal{X} \cup \mathcal{X}_\mathcal{I}$, we have two cases.

- In the first case, $u = \overline{w}^e\tau$ and then $\overline{v}^e\delta\tau\sigma = \overline{w}^e\tau\sigma$. If $v\delta \in \mathcal{C} \cup \mathcal{C}_{\mathcal{I}}$ then, $|\overline{v}^e\delta\tau\sigma| = 1$ which follows from the initial case of recurrence. Otherwise, $\exists w'$ such that $v\delta = f(w')$ where $f \in \mathcal{G}$. Let $F_0 = \{b\}$ if $v\delta = \{w'\}_b$ and $F_0 = \emptyset$ otherwise. Note that $F_0 \subseteq F$. Besides, since $\sigma \in [\![\, t \in Sub(v\delta, E, \mathcal{E}, \mathcal{K}) \,]\!]^e_{\tau'}$, then, $\exists u', \exists F'$ such that $u' \leqslant^{L'}_{F'} \overline{v\delta}^e\tau'$, $\overline{\mathcal{K}}^e\tau'\sigma \cap F'\sigma = \emptyset$, $F'\sigma \subseteq Dy(\overline{E}^e\tau'\sigma, \overline{\mathcal{K}}^e\tau'\sigma)$ and the two possibilities for u'. We have two cases: $u' = \overline{v\delta}^e\tau'$ or $u' <^{L'}_{F'} \overline{v\delta}^e\tau'$.
  In the first case, since $v\delta \notin \mathcal{X} \cup \mathcal{X}_{\mathcal{I}}$, $\overline{v\delta}^e\tau'\sigma = \overline{t}^e\tau'\sigma = \overline{t}^e\tau\sigma$. Then, $\overline{w}^e\tau\sigma = \overline{t}^e\tau\sigma$ and therefore it follows that $\overline{t}^e\tau\sigma \in Dy_d(\overline{E}^e\tau\sigma \cup \{\overline{w}^e\tau\sigma\}, \overline{\mathcal{K}}^e\tau\sigma)$.
  In the second case, $u' \leqslant^{L'}_{F'} \overline{w'}^e\tau' <^{L'}_{\overline{F_0}^e\tau'} \overline{v\delta}^e\tau'$. Then, $\sigma \in [\![\, t \in Sub(w', E, \mathcal{E}, \mathcal{K}) \,]\!]^e_{\tau'}$.
  Since $\left|\overline{w'}^e\tau'\sigma\right| < |\overline{w}^e\tau\sigma|$, it follows that $\overline{t}^e\tau'\sigma \in Dy_d(\overline{E}^e\tau\sigma \cup \{\overline{w'}^e\tau'\sigma\}, \overline{\mathcal{K}}^e\tau'\sigma)$.
  Note that $\overline{E}^e\tau' = \overline{E}^e\tau = \overline{E}^e$.
  Besides, $\overline{w'}^e\tau'\sigma \in Dy_d(\overline{E}^e\tau\sigma \cup \{\overline{w}^e\tau\sigma\}, \overline{\mathcal{K}}^e\tau\sigma)$ since $\overline{w'}^e\tau'\sigma <^{L'\sigma}_{\overline{F_0}^e\tau'\sigma} \overline{v\delta}^e\tau'\sigma$, $\overline{v\delta}^e\tau'\sigma = \overline{w}^e\tau\sigma, \overline{\mathcal{K}}^e\tau'\sigma \cap \overline{F_0}^e\tau'\sigma = \emptyset$ and $\overline{F_0}^e\tau'\sigma \subseteq Dy(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$ ($\overline{F_0}^e\tau' \subseteq F'$).
  Thus, $\overline{t}^e\tau\sigma \in Dy_d(\overline{E}^e\tau\sigma \cup \{\overline{w}^e\tau\sigma\}, \overline{\mathcal{K}}^e\tau\sigma)$.
- In the second case, $\overline{v}^e\delta\tau'\sigma <^{L\sigma}_{F\sigma} \overline{w}^e\tau\sigma$ and $\overline{t}^e\tau\sigma \in Dy_d(\overline{E}^e\tau\sigma \cup \{\overline{v}^e\delta\tau'\sigma\}, \overline{\mathcal{K}}^e\tau\sigma)$ since $|\overline{v}^e\delta\tau'\sigma| < |\overline{w}^e\tau\sigma|$. Besides, $\overline{v}^e\delta\tau'\sigma \in Dy_d(\overline{E}^e\tau\sigma \cup \{\overline{w}^e\tau\sigma\}, \overline{\mathcal{K}}^e\tau\sigma)$ thanks to the definition $\leqslant_{\_}$ and since $\overline{v}^e\delta\tau'\sigma <^{L\sigma}_{F\sigma} \overline{w}^e\tau\sigma$, $\overline{\mathcal{K}}^e\tau'\sigma \cap F\sigma = \emptyset$ and $F\sigma \subseteq Dy(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$. Thus, $\overline{t}^e\tau\sigma \in Dy_d(\overline{E}^e\tau\sigma \cup \{\overline{w}^e\tau\sigma\}, \overline{\mathcal{K}}^e\tau\sigma)$.

□

Let us go back to the proof of Proposition 5.
Let $\sigma \in \left|\left[\, (B \wedge (t \in Forge_c(E, \mathcal{K})) \vee \bigvee_{w \in E} (B \wedge t \in Sub(w, E, \mathcal{E}, \mathcal{K})) \,\right]\right|^e_\tau$.
If $\sigma \in [\![\, B \wedge t \in Forge_c(E, \mathcal{K}) \,]\!]^e_\tau$ then $\sigma \in [\![\, B \wedge t \in Forge(E, \mathcal{K}) \,]\!]^e_\tau$ by definition of $Forge$ and $Forge_c$. Let $\sigma \in \left|\left[\, \bigvee_{w \in E} B \wedge t \in Sub(w, E, \mathcal{E}, \mathcal{K}) \,\right]\right|^e_\tau$. Then, $\exists w \in E$ such that $\sigma \in [\![\, t \in Sub(w, E, \mathcal{E}, \mathcal{K}) \,]\!]^e_\tau \cap [\![\, B \,]\!]^e_\tau$. According to Lemma 8, $\overline{t}^e\tau\sigma \in Dy_d(\overline{E}^e\tau\sigma \cup \{\overline{w}^e\tau\sigma\}, \overline{\mathcal{K}}^e\tau\sigma)$. However, $\overline{w}^e\tau\sigma \in \overline{E}^e\tau\sigma$. Thus, $\overline{t}^e\tau\sigma \in Dy_d(\overline{E}^e\tau\sigma, \overline{t}^e\tau\sigma)$. Thus, Proposition 5 follows. □

Proposition 7. *Rules 10, 11 and 12 are correct and complete.*

Proof. Correctness of Rule 10 is trivial. For the completeness of Rule 10, let $\sigma \in [\![\, \langle t_1, \ldots, t_m\rangle \in Forge_c(E, \mathcal{K}) \,]\!]^e_\tau$. Then, $\langle \overline{t_1}^e\tau\sigma, \ldots, \overline{t_m}^e\tau\sigma\rangle \in Dy_c(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$ since we have $\overline{\langle t_1, \ldots, t_m\rangle}^e\tau\sigma \in Dy_c(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$. By definition of Dy_c, $\exists D$ a derivation with goal $\langle \overline{t_1}^e\tau\sigma, \ldots, \overline{t_m}^e\tau\sigma\rangle$ without using any term of $\overline{\mathcal{K}}^e\tau\sigma$ as a key for decryption and ending with a composition rule. Then, subterms of $\langle \overline{t_1}^e\tau\sigma, \ldots, \overline{t_m}^e\tau\sigma\rangle$ have to be in D, and therefore $\bigwedge_{i \leq m} (\overline{t_i}^e\tau\sigma \in Dy(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma))$. Finally, the proof of correctness and completeness of Rules 11 and 12 is similar to the one of Rule 10. □

In order to prove Proposition 8, we first prove Lemma 9.

Lemma 9. $[\![\, mpair(k, t) \in Forge_c(E, \mathcal{K}) \,]\!]^e_\tau = [\![\, \forall k\ t \in Forge(E, \mathcal{K}) \,]\!]^e_\tau$

Proof. Let $\sigma \in [\![\, mpair(k, t) \in Forge_c(E, \mathcal{K}) \,]\!]^e_\tau$. Then, $\overline{mpair(k, t)}^e\tau\sigma \in Dy_c(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$ leading to $\langle \overline{\tau_{k,1}(t)}^e\tau\sigma, \ldots, \overline{\tau_{k,e}(t)}^e$
Then, $\sigma \in [\![\, \forall k\ t \in Forge(E, \mathcal{K}) \,]\!]^e_\tau$ since $\bigwedge_{i \leq e} (\overline{\tau_{k,i}(t)}^e\tau\sigma \in Dy(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma))$. □

Proposition 8. *Rule 13 is correct and complete.*

Proof. According to our semantics, we have : $[\![\, \forall Q\ \exists R\ S \vee (B \wedge mpair(k, t) \in Forge(E, \mathcal{K})) \,]\!]^e_{\_}$
$= \bigcap_Q \bigcup_R [\![\, S \vee (B \wedge mpair(k, t) \in Forge(E, \mathcal{K})) \,]\!]^e_\tau$
$= \bigcap_Q \bigcup_R [\![\, S \,]\!]^e_\tau \cup ([\![\, B \,]\!]^e_\tau \cap [\![\, mpair(k, t) \in Forge(E, \mathcal{K}) \,]\!]^e_\tau)$
$= \bigcap_Q \bigcup_R [\![\, S \,]\!]^e_\tau \cup ([\![\, B \,]\!]^e_\tau \cap [\![\, \forall kt \in Forge(E, \mathcal{K}) \,]\!]^e_\tau)$ (according to Lemma 9)
$= \bigcap_Q \bigcup_R \bigcap_k [\![\, S \,]\!]^e_{\tau'} \cup ([\![\, B \,]\!]^e_{\tau'} \cap [\![\, t \in Forge(E, \mathcal{K}) \,]\!]^e_{\tau'})$ (since k fresh, $\tau' = \tau.[k \leftarrow n_k]$)
$= \bigcap_Q \bigcap_k \bigcup_R [\![\, S \,]\!]^e_{\tau'} \cup ([\![\, B \,]\!]^e_{\tau'} \cap [\![\, t \in Forge(E, \mathcal{K}) \,]\!]^e_\tau)$ (thanks to *mpair* autonomy)
$= \bigcap_{Q.k} \bigcup_R [\![\, S \vee (B \wedge t \in Forge(E, \mathcal{K})) \,]\!]^e_\tau$
$= [\![\, \forall Q.k\ \exists R\ S \vee (B \wedge t \in Forge(E, \mathcal{K})) \,]\!]^e_\tau$ □

Correctness and completeness of Rule 14 is trivial. Completeness and correctness of Rule 15 follow from Proposition 9 and Proposition 10.

Proposition 9. *Rule 15 is complete.*

Proof. Let $\sigma \in [\![t \in Sub(w, E, \mathcal{E}, \mathcal{K})]\!]^e_\tau$ and $W = [\![(t = w) \vee (t \in Sub_d(w, E, \mathcal{E}, \mathcal{K}))]\!]^e_\tau$. We show that either $\sigma \in [\![(t = w)]\!]^e_\tau$ if $w = \{v\}_b$ and $b \in \mathcal{K}$ or $\sigma \in W$ otherwise. We know that $\exists u,\ \exists F, L$ such that $u \leqslant^L_F \overline{w}^e\tau$, $\overline{\mathcal{K}}^e\tau\sigma \cap F\sigma = \emptyset$, $F\sigma \subseteq Dy(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$ and the two cases of the semantics of the *Sub*. We distinguish two cases depending whether ($w = \{v\}_b$ and $b \in \mathcal{K}$) or not:

- ($w = \{v\}_b$ and $b \in \mathcal{K}$). By definition of $<_{\_}$, we have $\overline{w}^e\tau \in L$ and $\overline{b}^e \in F$, However, $\overline{\mathcal{K}}^e\tau\sigma \cap F\sigma = \emptyset$. Thus, $u \leqslant^L_F \overline{w}^e\tau$ leads to $u = \overline{w}^e\tau$. We can suppose that $w \notin \mathcal{X} \cup \mathcal{X}_\mathcal{I}$, otherwise $u \leqslant^L_F w$ leads to $u <^L_F w$. Therefore, we only have the first case of the semantic of *Sub* that is $u\sigma = \overline{t}^e\tau\sigma$ which leads to $\overline{w}^e\tau\sigma = \overline{t}^e\tau\sigma$.
- ($w \neq \{v\}_b$ or $b \notin \mathcal{K}$). Then, we have the two cases of the semantics of *Sub*:
  - In the first case, $u\sigma = \overline{t}^e\tau\sigma$. Moreover, $u \leqslant^L_F \overline{w}^e\tau$. Then, either $u <^L_F \overline{w}^e\tau$ which leads to $\sigma \in [\![(t \in Sub_d(w, E, \mathcal{E}, \mathcal{K}))]\!]^e_\tau$ or $u = \overline{w}^e\tau$. In the last case, we have $u\sigma = \overline{w}^e\tau\sigma$. However, $u\sigma = \overline{t}^e\tau\sigma$. Then, $\overline{w}^e\tau\sigma = \overline{t}^e\tau\sigma$ which leads to $\sigma \in [\![(t = w)]\!]^e_\tau$. Thus, for both cases, $\sigma \in W$.
  - The second case is the same for the semantics of *Sub* or Sub_d. Therefore $\sigma \in W$ since $\sigma \in [\![t \in Sub_d(w, E, \mathcal{E}, \mathcal{K})]\!]^e_\tau$.

Thus Rule 15 is complete. □

Proposition 10. *Rule 15 is correct.*

Proof. There are two cases whether ($w = \{v\}_b$ and $b \in \mathcal{K}$) or not:

- Suppose ($w = \{v\}_b$ and $b \in \mathcal{K}$). Let $\sigma \in [\![(t = w)]\!]^e_\tau$. Then, $\overline{w}^e\tau\sigma = \overline{t}^e\tau\sigma$. Let $u = \overline{w}^e\tau$. We have $u \leqslant^\emptyset_\emptyset \overline{w}^e\tau$, $\emptyset \subseteq Dy(\overline{E}^e\tau\sigma, \emptyset)$ and $u\sigma = \overline{w}^e\tau\sigma = \overline{t}^e\tau\sigma$. Thus, $\sigma \in [\![(t \in Sub(w, E, \mathcal{E}, \mathcal{K}))]\!]^e_\tau$.
- Suppose ($w \neq \{v\}_b$ or $b \notin \mathcal{K}$). Let $\sigma \in [\![(t = w) \vee (t \in Sub_d(w, E, \mathcal{E}, \mathcal{K}))]\!]^e_\tau$. There are two cases:
  - $\sigma \in [\![(t = w)]\!]^e_\tau$. We show in a similar way as the first case ($w = \{v\}_b$ and $b \in \mathcal{K}$) that $\sigma \in [\![(t \in Sub(w, E, \mathcal{E}, \mathcal{K}))]\!]^e_\tau$.
  - $\sigma \in [\![(t \in Sub_d(w, E, \mathcal{E}, \mathcal{K}))]\!]^e_\tau$. By definition of Sub_d (the same as *Sub* with the difference that $u <^L_F \overline{w}^e\tau$ if $u\sigma = \overline{t}^e\tau\sigma$) and since if $u <^L_F \overline{w}^e\tau$ then we have $u \leqslant^L_F \overline{w}^e\tau$, then $\sigma \in [\![(t \in Sub(w, E, \mathcal{E}, \mathcal{K}))]\!]^e_\tau$.

Thus Rule 15 is correct. □

Proposition 11. *Rule 16 is correct and complete.*

Proof. Let $W = [\![\bigvee_{i=1\ldots m} (t \in Sub(t_i, E, \mathcal{E}, \mathcal{K}))]\!]^e_\tau$.

Let $\sigma \in [\![t \in Sub_d(\langle t_1, .., t_m\rangle, E, \mathcal{E}, \mathcal{K})]\!]^e_\tau$. Then, $\exists u,\ \exists F, L$ such that $u <^L_F \overline{\langle t_1, .., t_m\rangle}^e\tau$, $\overline{\mathcal{K}}^e\tau\sigma \cap F\sigma = \emptyset$, $F\sigma \subseteq Dy(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$ and the two possibilities for u (the two cases of the semantics of *Sub*). Then $\exists i = 1 \ldots m$. such that $u \leqslant^L_F \overline{t_i}^e\tau <^{\overline{\langle t_1,..,t_m\rangle}^e\tau}_\emptyset \overline{\langle t_1, .., t_m\rangle}^e\tau$ since $u <^L_F \overline{\langle t_1, .., t_m\rangle}^e\tau$. Thus, $\sigma \in W$.

Let $\sigma \in W$. Let $i = 1 \ldots m$ such that $\sigma \in [\![(t \in Sub(t_i, E, \mathcal{E}, \mathcal{K}))]\!]^e_\tau$. Then, $\exists u,\ \exists F, L$ such that $u \leqslant^L_F \overline{t_i}^e\tau$, $\overline{\mathcal{K}}^e\tau\sigma \cap F\sigma = \emptyset$, $F\sigma \subseteq Dy(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$ and the two possibilities for u (the two cases of the semantics of *Sub*). Moreover, since $t_i <_\emptyset \langle t_1, .., t_m\rangle$, then $\overline{t_i}^e\tau <^{\overline{\langle t_1,..,t_m\rangle}^e\tau}_\emptyset \overline{\langle t_1, .., t_m\rangle}^e\tau$. Then, $u <^{L\cup\{\overline{\langle t_1,..,t_m\rangle}^e\tau\}}_F \overline{\langle t_1, .., t_m\rangle}^e\tau$ and $\overline{\mathcal{K}}^e\tau\sigma \cap F\sigma = \emptyset$ which leads to $\sigma \in [\![t \in Sub_d(\langle t_1, .., t_m\rangle, E, \mathcal{E}, \mathcal{K})]\!]^e_\tau$. □

Proposition 12. *Rules 17 and 18 are correct and complete.*

Proof. First focus on Rule 17. Let $W = [\![t \in Sub(w, E, \mathcal{E}, \mathcal{K}) \wedge b \in Forge(E, \mathcal{K} \cup \{b\})]\!]^e_\tau$.

Let $\sigma \in [\![t \in Sub_d(\{w\}^s_b, E, \mathcal{E}, \mathcal{K})]\!]^e_\tau$. Then, $\exists u,\ \exists F, L$ s.t $u <^L_F \overline{\{w\}^s_b}^e\tau$, $\overline{\mathcal{K}}^e\tau\sigma \cap F\sigma = \emptyset$, $F\sigma \subseteq Dy(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$ and the two possibilities for u (the two cases of the semantics of *Sub*). Since $u <^L_F \overline{\{w\}^s_b}^e\tau$, then $u \leqslant^L_F \overline{w}^e\tau <^{\overline{\{w\}^s_b}^e\tau}_{\overline{b}^e\tau} \overline{\{w\}^s_b}^e\tau$. Thus, $\sigma \in [\![t \in Sub(w, E, \mathcal{E}, \mathcal{K})]\!]^e_\tau$. Moreover, since $u <^L_F \overline{\{w\}^s_b}^e\tau$, then $\overline{b}^e\tau \in F$, and therefore $\overline{b}^e\tau\sigma \in Dy(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$. By minimality of the derivation of goal $\overline{b}^e\tau\sigma$, $\overline{b}^e\tau\sigma$ has not to be used as a key for a decryption and then $\overline{b}^e\tau\sigma \in Dy(\overline{E}^e\tau\sigma, \overline{\mathcal{K} \cup \{b\}}^e\tau\sigma)$. Thus, $\sigma \in [\![b \in Forge(E, \mathcal{K} \cup \{b\})]\!]^e_\tau$. Therefore, $\sigma \in W$.

Let $\sigma \in W$. Let $\sigma \in [\![t \in Sub(w, E, \mathcal{E}, \mathcal{K})]\!]^e_\tau$. Then, $\exists u,\ \exists F, L$ such that $u \leqslant^L_F \overline{w}^e\tau$, $\overline{\mathcal{K}}^e\tau\sigma \cap F\sigma = \emptyset$, $F\sigma \subseteq Dy(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$ and the two possibilities for u (the two cases of the semantics of *Sub*). Besides, since $\overline{w}^e\tau <^{\{\overline{\{w\}^s_b}^e\tau\}}_{\overline{b}^e\tau} \overline{\{w\}^s_b}^e\tau$, we have $u <^{L\cup\{\overline{\{w\}^s_b}^e\tau\}}_{F\cup\{\overline{b}^e\tau\}} \overline{\{w\}^s_b}^e\tau$. However, $b \notin \mathcal{K}$ (du to the condition of

Rule 15). Then, $F\sigma \cap (\{\overline{b}^e\tau\sigma\} \cup \overline{\mathcal{K}}^e\tau\sigma) = \emptyset$. Moreover, since $\sigma \in [\![b \in Forge(E, \mathcal{K} \cup \{b\})]\!]^e_\tau$, we have $\overline{b}^e\tau\sigma \in Dy(\overline{E}^e\tau\sigma, \overline{\mathcal{K} \cup b}^e\tau\sigma)$. Then, $\overline{b}^e\tau\sigma \in Dy(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$, and therefore $F\sigma \cup \{\overline{b}^e\tau\sigma\} \subseteq Dy(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$. Thus, $\sigma \in [\![t \in Sub_d(\{w\}^s_b, E, \mathcal{E}, \mathcal{K})]\!]^e_\tau$.

Finally, correctness and completeness of Rule 18 is similar as Rule 17. □

Proposition 13. *Rule 19 is correct and complete.*

Proof. First focus on completeness. Let $\sigma \in [\![t \in Sub_d(mpair(k, w), E, \mathcal{E}, \mathcal{K})]\!]^e_\tau$. Then, $\exists u\ \exists F, L$ such that $u <^L_F \overline{mpair(k, w)}^e\tau$, $\overline{\mathcal{K}}^e\tau\sigma \cap F\sigma = \emptyset$, $F\sigma \subseteq Dy(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$ and the two possibilities for u. Therefore, $u <^L_F \langle \overline{\tau_{k,1}(w)}^e\tau\sigma, \ldots, \overline{\tau_{k,e}(w)}^e\tau\sigma\rangle$. Then, $\exists x \in \{1, \ldots, e\}$ s.t $u \leqslant^L_F \overline{\tau_{k,1}(w)}^e\tau\sigma$ which leads to $\sigma \in \bigcup_{x=1\ldots e} [\![t \in Sub(w, E, \mathcal{E}, \mathcal{K})]\!]^e_\tau$. Thus, $\sigma \in [\![\exists k\ (t \in Sub(w, E, \mathcal{E}, \mathcal{K}))]\!]^e_\tau$.

Second, focus on correctness. Let $\sigma \in [\![\exists k\ (t \in Sub(w, E, \mathcal{E}, \mathcal{K}))]\!]^e_\tau$.
Then, $\sigma \in \bigcup_{x=1\ldots e} [\![t \in Sub_d(w, E, \mathcal{E}, \mathcal{K})]\!]^e_{\tau,[k\leftarrow x]}$. Let $x \in \{1, \ldots, e\}$ and $\tau' = \tau, [k \leftarrow x]$. We have $\sigma \in [\![t \in Sub_d(w, E, \mathcal{E}, \mathcal{K})]\!]^e_{\tau'}$. Then, $\exists u\ \exists F, L$ such that $u \leqslant^L_F \overline{w}^e\tau'$, $\overline{\mathcal{K}}^e\tau\sigma \cap F\sigma = \emptyset$, $F\sigma \subseteq Dy(\overline{E}^e\tau'\sigma, \overline{\mathcal{K}}^e\tau'\sigma)$ and the two possibilities for u. However, $\overline{w}^e\tau' = \overline{\tau_{k,x}(w)}^e\tau$. Then, $u \leqslant^L_F \overline{\tau_{k,x}(w)}^e\tau <^{\langle \overline{\tau_{k,1}(w)}^e\tau, \ldots, \overline{\tau_{k,e}(w)}^e\tau\rangle}_\emptyset \langle \overline{\tau_{k,1}(w)}^e\tau, \ldots, \overline{\tau_{k,e}(w)}^e\tau$
Thus, $u <^{L \cup \{\langle \overline{\tau_{k,1}(w)}^e\tau\sigma, \ldots, \overline{\tau_{k,e}(w)}^e\tau\rangle\}}_F \overline{mpair(k, w)}^e\tau$ and $\overline{\mathcal{K}}^e\tau\sigma \cap F\sigma = \emptyset$ which leads to $\sigma \in [\![t \in Sub_d(mpair(k, w), E, \mathcal{E}, \mathcal{K})]\!]^e_\tau$. □

Proposition 14. *Rule 20 is correct and complete.*

Proof. Correctness of Rule 20 is trivial. Let $\sigma \in [\![t \in Sub_d(c, E, \mathcal{E}, \mathcal{K})]\!]^e_\tau$ where $c \in \mathcal{C} \cup \mathcal{C}_\mathcal{I}$. Then, $\exists u\ \exists F, L$ such that $u <^L_F \overline{c}^e\tau$, $\overline{\mathcal{K}}^e\tau\sigma \cup F\sigma = \emptyset$, $F\sigma \subseteq Dy(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$ and the two possibilities for u. In the first case, $u\sigma = \overline{t}^e\tau\sigma$ and $u <^L_F \overline{c}^e\tau$. However, the last condition is impossible since $c\tau \in \mathcal{C} \cup \mathcal{C}_\mathcal{I}$ ($\overline{c}^e\tau = c\tau$), and therefore $u = c\tau$. In the same way, the second case of Sub treats the case where $u \in \mathcal{X} \cup \mathcal{X}_\mathcal{I}$ which is in contradiction with $u = c\tau \in \mathcal{C} \cup \mathcal{C}_\mathcal{I}$. We conclude that in the two cases, we have $\sigma \in [\![\bot]\!]^e_\tau$. □

Proposition 15. *Rule 23 is correct and complete.*

Proof. This follows directly from our semantics: $\sigma \in [\![f(u_1, .., u_m) = f(w_1, .., w_m)]\!]^e_\tau$ iff $\overline{f(u_1, .., u_m)}^e\tau\sigma = \overline{f(w_1, .., w_m)}^e\tau\sigma$ iff $f(\overline{u_1}^e\tau\sigma, .., \overline{u_m}^e\tau\sigma) = f(\overline{w_1}^e\tau\sigma, .., \overline{w_m}^e\tau\sigma)$ iff $\bigwedge_{i=1\ldots m}(\overline{u_i}^e\tau\sigma = \overline{w_i}^e\tau\sigma)$ iff $\sigma \in [\![\bigwedge_{i=1\ldots m} u_i = w_i]\!]^e_\tau$. □

In order to prove Proposition 16, we first prove Lemma 10.

Lemma 10. $[\![mpair(k, u) = mpair(l, w)]\!]^e_\tau = [\![\forall k\, u = w\delta_{l,k}]\!]^e_\tau$

Proof.

$\sigma \in [\![mpair(k, u) = mpair(l, w)]\!]^e_\tau$	iff	$\overline{mpair(k, u)}^e\tau\sigma = \overline{mpair(l, w)}^e\tau\sigma$	iff
$\langle \overline{\tau_{k,1}(u)}^e, \ldots, \overline{\tau_{k,v}(u)}^e\rangle\tau\sigma = \langle \overline{\tau_{l,1}(w)}^e, \ldots, \overline{\tau_{l,v}(w)}^e\rangle\tau\sigma$	iff		
$\langle \overline{\tau_{k,1}(u)}^e\tau\sigma, \ldots, \overline{\tau_{k,v}(u)}^e\tau\sigma\rangle = \langle \overline{\tau_{l,1}(w)}^e\tau\sigma, \ldots, \overline{\tau_{l,v}(w)}^e\tau\sigma\rangle$	iff		
$\bigwedge_{i=1,\ldots,e}(\overline{\tau_{k,i}(u)}^e\tau\sigma = \overline{\tau_{l,i}(w)}^e\tau\sigma)$	iff	$\bigwedge_{i=1,\ldots,e}(\overline{\tau_{k,i}(u)}^e\tau\sigma = \overline{\tau_{k,i}(w\delta_{l,k})}^e\tau\sigma)$	iff
$\bigwedge_{i=1,\ldots,e}(\overline{u}^e\tau\tau_{k,i}\sigma = \overline{w\delta_{l,k}}^e\tau\tau_{k,i}\sigma)$	iff	$\sigma \in \bigcap_{i=1,\ldots,e} [\![u = w\delta_{l,k}]\!]^e_{\tau,\tau_{k,i}}$	iff
$\sigma \in [\![\forall k\, u = w\delta_{l,k}]\!]^e_\tau$			

□

Proposition 16. *Rule 24 is correct and complete.*

Proof. According to our definitions, we have: $[\![\forall Q\ \exists R\ S \vee (B \wedge (mpair(k, u) = mpair(l, w)))]\!]^e_\_$
$= \bigcap_Q \bigcup_R [\![S \vee (B \wedge (mpair(k, u) = mpair(l, w)))]\!]^e_\tau$ (where τ assigns values to Q and R)
$= \bigcap_Q \bigcup_R ([\![S]\!]^e_\tau \cup ([\![B]\!]^e_\tau \cap [\![mpair(k, u) = mpair(l, w)]\!]^e_\tau))$
$= \bigcap_Q \bigcup_R ([\![S]\!]^e_\tau \cup ([\![B]\!]^e_\tau \cap [\![\forall k\, u = w\delta_{l,k}]\!]^e_\tau))$ (according to Lemma 10)
$= \bigcap_Q \bigcup_R \bigcap_k ([\![S]\!]^e_{\tau'} \cup ([\![B]\!]^e_{\tau'} \cap [\![u = w\delta_{l,k}]\!]^e_{\tau'}))$ (since k is fresh, $\tau' = \tau.[k \leftarrow n_k]$)
$= \bigcap_Q \bigcap_k \bigcup_R ([\![S]\!]^e_{\tau'} \cup ([\![B]\!]^e_{\tau'} \cap [\![u = w\delta_{l,k}]\!]^e_{\tau'}))$ (thanks to *mpair* autonomy, $Var_\mathcal{I}(u) \cap R = Var_\mathcal{I}(w) \cap R = \emptyset$))
$= \bigcap_{Q.k} \bigcup_R ([\![S \vee (B \wedge (u = w\delta_{l,k}))]\!]^e_{\tau'}) = [\![\forall Q.k\ \exists R\ S \vee (B \wedge (u = w\delta_{l,k}))]\!]^e_\tau$. □

Proposition 17. *Rule 25 is correct and complete.*

Proof. This follows directly from the fact that each two constants c_i and c_j are different. □

Proposition 18. *Rules 26 and 27 are correct and complete.*

Proof. Correctness of Rule 26 is trivial. For it's completeness, let $\sigma \in \left|\!\left[B \wedge (X_i = u)^{\bullet} \wedge (X_i = v)^{\bullet} \right]\!\right|_{\tau}^{e}$. Then, $X_i\tau\sigma = \overline{u}^e\tau\sigma$ and $X_i\tau\sigma = \overline{v}^e\tau\sigma$. Thus, $\sigma \in \left|\!\left[B \wedge (X_i = u)^{\bullet} \wedge (X_i = v)^{\bullet} \wedge u = v \right]\!\right|_{\tau}^{e}$ since $\overline{u}^e\tau\sigma = \overline{v}^e\tau\sigma$. Also, correctness and completeness of Rule 27 is similar as Rule 26. □

Proposition 19. *Rules 28, 29 and 30 are correct and complete.*

Proof. First focus on Rule 28. According to our semantics, we have:

$\sigma \in \left|\!\left[(X_i = u)^{\bullet} \wedge X_i \in Forge_c(E', \mathcal{K}) \right]\!\right|_{\tau}^{e}$ iff $\sigma \in \left|\!\left[(X_i = u)^{\bullet} \right]\!\right|_{\tau}^{e} \cap [\![X_i \in Forge_c(E', \mathcal{K})]\!]_{\tau}^{e}$ iff
$X_i\tau\sigma = \overline{u}^e\tau\sigma$ and $X_i\tau\sigma \in Dy_c(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$ iff $X_i\tau\sigma = \overline{u}^e\tau\sigma$ and $\overline{u}^e\tau\sigma \in Dy_c(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$ iff
$\sigma \in \left|\!\left[(X_i = u)^{\bullet} \wedge u \in Forge_c(E', \mathcal{K}) \right]\!\right|_{\tau}^{e}$.

Second, correctness and completeness of Rule 29 follows in a similar way. Moreover, this is trivial for Rule 30 as it follows direclty from the fact that $E \subset E'$ and since $A\tau\sigma \in Dy_c(\overline{E}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$, then $A\tau\sigma \in Dy_c(\overline{E'}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$. □

Proposition 20. *Rule 31 is correct and complete.*

Proof. correctness: Let $\sigma \in [\![(X = w)^{sm} \wedge t \in Sub(w, E', \mathcal{E}, \mathcal{K}) \wedge X \notin Forge_c(E, \mathcal{K})]\!]_{\tau}^{e}$. Let $u = X$, $\delta = \emptyset$, $\tau' = \tau$. Then, $u\sigma = X\sigma = \overline{w}^e\delta\tau'\sigma$ that leads to $u\sigma \notin Dy_c(\overline{E'}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$, and $\sigma \in [\![t \in Sub(w\delta, E', \mathcal{E}, \mathcal{K})]\!]_{\tau'}^{e}$. Moreover, $u <_{\emptyset}^{\emptyset} X$ and $\emptyset \subseteq Dy(\overline{E'}^e\sigma, \emptyset)$, thus proving $\sigma \in [\![t \in Sub_d(X, E', \mathcal{E}, \mathcal{K})]\!]_{\tau}^{e}$ according to the definition.

Completeness: let $\sigma \in [\![(X = w)^{sm} \wedge t \in Sub_d(X, E', \mathcal{E}, \mathcal{K})]\!]_{\tau}^{e}$ where $(X = w)^{sm} \in \mathcal{E}$. Let u, F, L be the objects defined by $u \leqslant_F^L X$, $F\sigma \cap \overline{\mathcal{K}}^e\tau\sigma = \emptyset$ and $F\sigma \subseteq Dy(\overline{E'}^e\sigma, \overline{\mathcal{K}}^e\sigma)$ in the definition of $t \in Sub_d(X, E', \mathcal{E}, \mathcal{K})$. Since Sub_d ensures that $u <_F^L X$ in case $u\sigma = \overline{t}^e\tau\sigma$ and since $u <_F^L X$ is impossible, then we have $u\sigma \neq \overline{t}^e\tau\sigma$ and $u = X$. Thus, u and F validates the second case of the semantics of Sub with $u \in \mathcal{X}$, and therefore, $\exists v, \delta, k, \tau'$ such that $(X = v) \in \mathcal{E}$, $\delta = \emptyset$, $\tau' = \tau$, $k, i \notin \{t, X, \mathcal{E}\}$, $X \notin Forge_c(E, \mathcal{K})$, $u\sigma = \overline{v}^e\delta\tau'\sigma$ and $\sigma \in [\![t \in Sub(v\delta, E', \mathcal{E}, \mathcal{K})]\!]_{\tau}^{e}$. However, since $(X = u)^{sm} \in \mathcal{E}$, then we choose $v = w$ which lead to $\sigma \in [\![(X = w)^{sm} \wedge t \in Sub(w, E', \mathcal{E}, \mathcal{K}) \wedge X \notin Forge_c(E, \mathcal{K})]\!]_{\tau}^{e}$. □

Proposition 21. *Rule 32 is correct and complete.*

Proof. Correctness of Rule 32 is trivial, so we focus on completeness.
Let $\sigma \in [\![A \in Forge_c(E, \mathcal{K}) \wedge t \in Sub_d(A, E', \mathcal{E}, \mathcal{K})]\!]_{\tau}^{e}$. Let u, F be such that $u \leqslant_F^L A$ and $F\sigma \subseteq Dy(\overline{E'}^e\sigma, \overline{\mathcal{K}}^e\sigma)$. Since Sub_d ensures that $u <_F^L A$ in case $u\sigma = \overline{t}^e\tau\sigma$ and since $u <_F^L A$ is impossible, then we have $u\sigma \neq \overline{t}^e\tau\sigma$ and $u = A$. Thus, u and F validates the second case of the semantics of Sub and therefore, $\exists v, \delta, k, \tau'$ such that $u\sigma \notin Dy_c(\overline{E'}^e\tau'\sigma, \overline{\mathcal{K}}^e\tau'\sigma)$. Nevertheless, $u\sigma = A\sigma$. Then, $A\sigma \notin Dy_c(\overline{E'}^e\sigma, \overline{\mathcal{K}}^e\tau'\sigma)$. Moreover, $A\sigma \in Dy_c(\overline{E}^e\sigma, \overline{\mathcal{K}}^e\tau'\sigma)$. However, $E \subseteq E'$. Indeed, according to Invariant 3, $\mathcal{L}(A) \subseteq E'$. Then, either $\mathcal{L}(A) = E$ or $\mathcal{L}(A) \subseteq E$. In the last case, there are two possibilities. In the first one, $A \in Forge_c(E, \mathcal{K})$ would be simplified to $A \in Forge_c(\mathcal{L}(A), \mathcal{K})$ if at level $\mathcal{L}(A)$ we have $A \in Forge_c(\mathcal{L}(A), \mathcal{K})$ as constraint for A. In the second possibility, $A \in Forge_c(E, \mathcal{K})$ would be eliminated if at level $\mathcal{L}(A)$ we have $A = u$ as constraint for A. This is due to the fact that for the level $\mathcal{L}(A)$ we have a constraint $Forge$ or $Equality$ according to the Proposition 1. We conclude that $E \subseteq E'$. Therfore, $A\sigma \notin Dy_c(\overline{E'}^e\sigma, \overline{\mathcal{K}}^e\tau'\sigma)$ and $A\sigma \in Dy_c(\overline{E'}^e\sigma, \overline{\mathcal{K}}^e\tau'\sigma)$ proving that $\sigma \in [\![\bot]\!]_{\tau}^{e}$. □

Proposition 22. *Rule 33 is correct and complete.*

Proof. Completeness: if $\sigma \in [\![t \in Sub_d(X_m, E', \mathcal{E}, \mathcal{K})]\!]_{\tau}^{e}$, then $\exists (X_j = v) \in \mathcal{E}$, $\exists k \notin \{t, X_m, \mathcal{E}\}$, $\exists n_k$ s.t. $\tau' = \tau.[k \leftarrow n_k]$, $\sigma \in [\![t \in Sub(v\delta, E', \mathcal{E}, \mathcal{K})]\!]_{\tau'}^{e}$, $\sigma \in [\![X_m = v\delta]\!]_{\tau'}^{e}$ and $\sigma \notin [\![X_m \in Forge_c(E', \mathcal{K})]\!]_{\tau'}^{e}$, with $\delta = \delta_{j,m}^{k}$. To show this, let u, F, L be the $u \leqslant_F^L X_m\tau$, $F\sigma \cap \overline{\mathcal{K}}^e\tau\sigma = \emptyset$ and $F\sigma \subseteq Forge(\overline{E'}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$ from the semantics of Sub_d proving that $\sigma \in [\![t \in Sub_d(X_m, E', \mathcal{E}, \mathcal{K})]\!]_{\tau'}^{e}$. Note that Sub_d is more restrictive than Sub in the sence that it ensures $u <_F^L X_m\tau$ in case $u\sigma = \overline{t}^e\tau\sigma$. However $u <_F^L X_m\tau$ is impossible, so $u\sigma \neq \overline{t}^e\tau\sigma$ and $u = X_m\tau$ necessarily. This means that u, F validate the second case of the semantics of Sub, and thus that $\exists v, \delta, k, i, j, \tau'$ s.t. $k, i \notin \{t, X_m, \mathcal{E}\}$, $\mathcal{D}om(\tau') = \mathcal{D}om(\tau) \cup \{k, i\}$, $u = X_i\tau'$, $(X_j = v) \in \mathcal{E}$, $\delta = \delta_{j,i}^{k}$, $u\sigma \notin Forge_c(\overline{E'}^e\tau\sigma, \overline{\mathcal{K}}^e\tau\sigma)$, $u\sigma = \overline{v}^e\delta\tau'\sigma$, and $\sigma \in [\![t \in Sub(v\delta, E', \mathcal{E}, \mathcal{K})]\!]_{\tau'}^{e}$. We remark that $\tau(m) = \tau'(i)$ since $X_m\tau = u = X_i\tau'$, and so the proposition follows by replacing i by m.

Correctness: if $\exists (X_j = v) \in \mathcal{E}$, $\exists k \notin \{t, X_m, \mathcal{E}\}$, $\exists n_k$ s.t. $\tau" = \tau.[k \leftarrow n_k]$, $\sigma \in [\![t \in Sub(v\delta, E', \mathcal{E}, \mathcal{K})]\!]_{\tau"}^{e}$, $\sigma \in [\![X_m = v\delta]\!]_{\tau"}^{e}$ and $\sigma \notin [\![X_m \in Forge_c(E', \mathcal{K})]\!]_{\tau"}^{e}$, with $\delta = \delta_{j,m}^{k}$, then $\sigma \in [\![t \in Sub_d(X_m, E', \mathcal{E}, \mathcal{K})]\!]_{\tau}^{e}$.

Assume that the precondition above is true for the objects $X_j, v, \mathcal{E}, k, \tau" = \tau.[k \leftarrow n_k]$. Let $u = X_m\tau$, thus $u\sigma = X_m\tau"\sigma$, and let i be a fresh index, $\delta' = \delta^k_{j,i}$, and let $\tau' = \tau".[i \leftarrow \tau(m)]$. Then we have $u = X_i\tau'$, $(X_j = v) \in \mathcal{E}$, $k, i \notin \{t, X_m, \mathcal{E}\}$, $u\sigma = \overline{v}^e\delta'\tau'\sigma$, $u\sigma \notin Forge_c(\overline{E'}^e\sigma, \overline{\mathcal{K}}^e\sigma)$ and $\sigma \in [\![\, t \in Sub(v\delta', E', \mathcal{E}, \mathcal{K}) \,]\!]^e_{\tau'}$. Moreover, we have $u \leqslant^{\emptyset}_{\emptyset} X_m\tau$ and $\emptyset \subseteq Forge(\overline{E}^e\sigma, \overline{\mathcal{K}}^e\sigma)$, thus proving $\sigma \in [\![\, t \in Sub_d(X_m, E', \mathcal{E}, \mathcal{K}) \,]\!]^e_\tau$ according to the definition. □

Proposition 23. *Rule 34 is correct and complete.*

Proof. We first focus on completeness, i.e. assume that $\sigma \in [\![\, X_m \in Forge_c(E', \mathcal{K}) \,]\!]^e_\tau$, and assume we have $\mathcal{M}(\overrightarrow{X})$ defined as in Rule 34. We denote by *Rhs* the right-hand side of Rule 34. Then we have two cases :

- Either $\exists o = 1..p\ \exists n_{k_o}$ s.t. $\sigma \in [\![\, X_m = u_o\delta_o \,]\!]^e_{\tau'}$ with $\tau' = \tau.[k'_o \leftarrow n_{k_o}]$. Then, we have $\overline{u_o}^e\delta_o\tau'\sigma = X_m\tau'\sigma$ which leads to $\overline{u_o}^e\delta_o\tau'\sigma \in Dy_c(\overline{E'}^e\sigma, \overline{\mathcal{K}}^e\sigma)$ and thus, $\sigma \in [\![\, Rhs \,]\!]^e_\tau$;
- Or there exists no such o, i.e. $\forall o = 1..p$, $\forall n_{k_o}$, $\sigma \notin [\![\, X_m = u_o\delta_o \,]\!]^e_{\tau'}$ with $\tau' = \tau.[k'_o \leftarrow n_{k_o}]$ i.e. $\overline{u_o}^e\delta_o\tau'\sigma \neq X_m\tau'\sigma$, and thus $\sigma \in \left|\left[\bigwedge_{o=1..p} \forall k'_o\, X_m \neq u_o\delta_o \right]\right|^e_\tau$. Then, $\sigma \in [\![\, Rhs \,]\!]^e_\tau$;

Second, for the correctness let *Rhs* be the right-hand side of Rule 34, and assume $\mathcal{M}(\overrightarrow{X})$ defined as in the rule. Assume also that $\sigma \in [\![\, Rhs \,]\!]^e_\tau$ because the second part of *Rhs* is validated (otherwise, the proposition is trivial). It means that $\exists o = 1..p$, $\exists n_{k_o}$ s.t. $\overline{u_o}^e\delta_o\tau' \in Dy_c(\overline{E'}^e\sigma, \overline{\mathcal{K}}^e\sigma)$ and $X_m\tau'\sigma = \overline{u_o}^e\delta_o\tau'\sigma$ with $\tau' = \tau.[k'_o \leftarrow n_{k_o}]$. It follows that $X_m\tau\sigma \in Dy_c(\overline{E'}^e\sigma, \overline{\mathcal{K}}^e\sigma)$ since $X_m\tau = X_m\tau'$. □

Proposition 24. *Rule 35 is correct and complete.*

Proof. Correctness of Rule 35 is trivial. For completeness, assume that $\sigma \in [\![\, X_m = v \,]\!]^e_\tau$, i.e. $X_m\tau\sigma = \overline{v}^e\tau\sigma$, and $\mathcal{M}(\overrightarrow{X})$ according to the definitions of Rule 35. Let us write $F = (\forall Q \exists R\, B_1 \vee .. \vee B_s)$ the whole formula in which Rule 35 is used, and assume that $\sigma \in [\![\, F \,]\!]^e$. We denote by *Rhs* the right-hand side of Rule 35. Then we have two cases :

- Either $\exists o = 1..p\ \exists n_{k_o}$ s.t. $\sigma \in [\![\, X_m = u_o\delta_o \,]\!]^e_{\tau'}$ with $\tau' = \tau.[k'_o \leftarrow n_{k_o}]$. Then, we have $\overline{u_o}^e\delta_o\tau'\sigma = \overline{v}^e\tau'\sigma$ which leads to $\sigma \in [\![\, Rhs \,]\!]^e_\tau$;
- Or there exists no such o, i.e. $\forall o = 1..p$, $\forall n_{k_o}$, $\sigma \notin [\![\, X_m = u_o\delta_o \,]\!]^e_{\tau'}$ with $\tau' = \tau.[k'_o \leftarrow n_{k_o}]$ i.e. $\overline{u_o}^e\delta_o\tau'\sigma \neq X_m\tau'\sigma$, and thus $\sigma \in \left|\left[\bigwedge_{o=1..p} \forall k'_o\, X_m \neq u_o\delta_o \right]\right|^e_\tau$. Moreover, thanks to corollary 1, we know that for any block in $B_1 \vee .. \vee B_s$, there exists a master constraint for $\overrightarrow{X}$ in this block. Besides, $\exists \tau"$ with $\forall i \in Q\ \tau"(i) = \tau'(m)$ such that $\exists j\ \sigma \in [\![\, B_j \,]\!]^e_{\tau"}$. Thus, σ validates at last one of the master constraints for $\overrightarrow{X}$ for $\tau"$. However, by hypothesis $\forall o = 1..p$, $\sigma \notin [\![\, X_m \neq u_o\delta_o \,]\!]^e_{\tau"}$. Thus, $\exists r = 1..q$ such that $\sigma \in [\![\, X_{j_r} \in Forge_c(E'_r, \mathcal{K}_r) \,]\!]^e_{\tau"}$. Since $\tau"(j_r) = \tau'(m) = \tau(m)$, it follows that $\sigma \in [\![\, Rhs \,]\!]^e_\tau$.

□

7 Termination For Protocols Without *mpair*'s and Without Indexed Variables

This section aims at proving the termination of rules presented in Section 4 for protocols without $mpair(,)$ and without index variables. Note that for autonomous protocols without *mpair*(,s), a constraint system would contain neither quantifiers, nor indexed variables and consequently no master constraints.

We first define a weight $\|\_\|$ for terms, elementary constraints, blocks and constraint systems. Then, we show that each rule decreases this weight.

In order to define the weight of a term, we need to introduce some definitions:

Definition 28 (Row of a term)**.** *A row of a term t is defined as follows:*

- $r(X) = max\{l \mid X \sqsubset_l Y, Y \in \mathcal{X}\}$ *for* $X \in \mathcal{X}$
- $r(t) = max\{r(Y) \mid Y < t, Y \in \mathcal{X}\}$

Definition 29 (Size of a term t)**.** *We define the size of a term t denoted by $|t|$ as follows:*

- $|t| = 1$ *for* $t \in \mathcal{X} \cup \mathcal{C}$
- $|f(u_1, .., u_m)| = 1 + |u_1| + .. + |u_m|$
- $|h(u)| = 2 + |u|$ *for* $h \in H$

We extend this definition to sets of terms by: $|E| = \Sigma_{t \in E}|t|$ *for* $E \subset T$.

Now, we can define the weight for terms, then, for elementary constraints and finally for constraint blocks and constraint system.

Definition 30 (Weight of a term t)**.** *Let p be the size of the protocol, i.e. the sum of sizes of messages. We define the weight of a term t denoted by $\|t\|$ as follows:*

- $\|X\| = p^{\mathrm{r}(X)+1}$ *for* $X \in \mathcal{X}$
- $\|c\| = 1$ *for* $c \in \mathcal{C}$
- $\|f(u_1, .., u_m)\| = 1 + \|u_1\| + .. + \|u_m\|$
- $\|h(u)\| = 2 + \|u\|$ *for* $h \in H$

We extend this definition in the natural way to sets of terms: $\|E\| = \Sigma_{t \in E}\|t\|$ *for* $E \subset T$.

Definition 31 (Weight of an elementary constraint)**.** *Let st be the number of the protocol subterms. We define the weight of an elementary constraint ctr denoted by $\|ctr\|$ as follows:*

- $\|t \in Forge(E, \mathcal{K})\| = \langle st - \#\mathcal{K},\ \|t\| + \|E\| + |E| + 1\rangle$
- $\|t \in Forge_c(E, \mathcal{K})\| = \langle st - \#\mathcal{K},\ \|t\| + \|E\| + |E|\rangle$
- $\|t \in Sub(w, E, \mathcal{E}, \mathcal{K})\| = \langle st - \#\mathcal{K},\ \|t\| + \|w\| + |E| + 1\rangle$
- $\|t \in Sub_d(w, E, \mathcal{E}, \mathcal{K})\| = \langle st - \#\mathcal{K},\ \|t\| + \|w\| + |E|\rangle$
- $\|t = u\| = \langle 0,\ \|t\| + \|u\|\rangle$

Definition 32 (Weight of a constraint block)**.** *Let NcE be the maximum number of* Equality *constraints of the form $X = u$ where $X \in X$ and NcN be the maximum number of* Negative *constraints of type $Forge$. Consider a constraint block: $B = ctr_1 \wedge ..ctr_l$. We denote by Nc_B the number of of negative constraints in B, Sc_B the number of submaster constraints in B and by Ec_B the number of equality constraints of the form $X = u$ ($X \in \mathcal{X}$) in B. We denote also the lexicographic order by $\langle\rangle$ and the multiset by $[]$. We define the weight of B as follows:*

$$\|B\| = \left\langle NcN - Nc_B,\ NcE - Sc_B,\ [\|ctr_i\|]_{ctr_i \in B},\ NcE - Ec_B \right\rangle$$

Definition 33 (Weight of a constraint System)**.** *Consider a constraint system: $S = B_1 \wedge ..B_l$. We define the weight of S as follows:*

$$\|S\| = [\|B_i\|]_{B_i \in S}$$

Proposition 25. *Algorithm 1 terminates for protocols without $mpair(,)$ and without index variables.*

Proof. Proof. We prove Proposition 25 by showing that each rule of Section 4 which is related to non indexed variables decreases the weight of the whole constraint system S. That is, considering a rule r, $\|\mathrm{post}(r)\| < \|\mathrm{pre}(r)\|$. First, note that our rules do not eliminate negative constraints. Then, $\forall B \in S$, $NcN - Nc_B$ either remains unchanged or decreases for the case of Rule 31. Thus, Rule 31 decreases $\|S\|$. Rules 3, 14, 20, 22, 32 eliminate one block of the constraint system since they lead to $\bot$. Then, they decrease $\|S\|$. Rule 22 eliminates one constraint of a certain block of S since they lead to $\top$. Then, it decreases $\|S\|$. Rule 1 changes an equality constraint $ctr \in B$ into another one of the form $(X = u)$ where $X \in X$. Both the number of submaster constraints and the weight of constraints in B remain unchanged. Besides, $NcE - Ec_B$ decreases since Ec_B increases. Then, $\|B\|$ decreases and so does $\|S\|$.
Rule 8 adds a submaster constraint to a block $B \in S$. Then, $\|S\|$ decreases.
For Rule 2, since $(X = u)^{sm} \in B$, $\|X\| > \|u\|$. Indeed, $\|u\| \leq |u| * p^{max\{\mathrm{r}(Y)|Y<u, Y \in \mathcal{X}\}} \leq p^{\mathrm{r}(X)+1}$. Then, $\|Y = X\| = \|Y\| + \|X\| > \|Y\| + \|u\| = \|Y = u\|$. Besides, the number of submaster constraints remains unchanged. Thus, $\|S\|$ decreases.

For Rule 9, $\|\text{pre}(R\ 9)\| > \|\text{post}(R\ 9)\|$. Indeed, $\|\text{pre}(R\ 9)\| = \langle k, \|t\| + \|E\| + |E| + 1\rangle$ and $\|\text{post}(R\ 9)\| = [\langle k, \|t\| + \|E\| + |E|\rangle, \langle k, \|t\| + \|w\| + |E| + 1\rangle]$, where $k = st - \#\mathcal{K}$ and $w \in E$. Moreover, Rule 9 do not change the number of submaster constraints. Then, $\|S\|$ decreases.
For Rule 10, $\|\text{pre}(R\ 10)\| = \langle k, 1 + \|t_1\| + .. + \|t_m\| + \|E\| + |E|\rangle$ and $\|\text{post}(R\ 10)\| = [\langle k, \|t_i\| + \|E\| + |E| + 1\rangle]_{t_i < \langle t_1..t_m\rangle}$, where $k = st - \#\mathcal{K}$. Moreover, Rule 10 do not change the number of submaster constraints. Then, $\|\text{pre}(R\ 10)\| > \|\text{post}(R\ 10)\|$. Therefore, $\|B\|$ decreases and so does $\|S\|$.
We show in a similar way as Rule 10 that for Rule 11, $\|S\|$ decreases.
For Rule 12, $\|\text{pre}(R\ 12)\| > \|\text{post}(R\ 12)\|$. Indeed, $\|\text{pre}(R\ 12)\| = \langle k, 2 + \|t\| + \|E\| + |E|\rangle$ and $\|\text{post}(R\ 12)\| = \langle k, 1 + \|t\| + \|E\| + |E|\rangle$, where $k = st - \#\mathcal{K}$. Moreover, Rule 12 do not change the number of submaster constraints. Then, $\|B\|$ decreases and so does $\|S\|$.
For Rule 15, we have $\|\text{pre}(R\ 15)\| = \langle k, 1 + \|t\| + \|u\| + |E|\rangle$ and two cases for post$(R\ 15)$. In the first one, $\|\text{post}(R\ 15)\| = \langle k, \|t\| + \|u\|\rangle$. In the second case, $\|\text{post}(R\ 15)\| = [\langle k, \|t\| + \|u\|\rangle, \langle k, \|t\| + \|u\| + |E|\rangle]$. In both the two cases, $\|\text{pre}(R\ 15)\| > \|\text{post}(R\ 15)\|$. Besides, since this rule do not modify the number of submaster constraints, $\|S\|$ decreases.
We show in a similar way as Rule 10 that for Rule 16, $\|S\|$ decreases.
For Rule 17, first, $\|\text{pre}(R\ 17)\| = \langle k, 1 + \|t\| + \|u\| + \|b\| + |E|\rangle$ where $k = st - \#\mathcal{K}$. Second, $\|\text{post}(R\ 17)\| = [\langle k, 1 + \|t\| + \|u\| + |E|\rangle, \langle k', 1 + \|b\| + \|E\| + |E|\rangle]$ where $k' = st - \#\mathcal{K} \cup \{\{u\}_p^p\}$. Since $k' < k$, $\|\text{pre}(R\ 17)\| > \|\text{post}(R\ 17)\|$.
We show in a similar way as Rule 17 that for Rule 18, $\|S\|$ decreases.
For Rule 23, first, $\|\text{pre}(R\ 23)\| = \langle k, 2 + \|u_1\| + .. + \|u_m\| + \|w_1\| + .. + \|w_m\|\rangle$. Second, $\|\text{pre}(R\ 23)\| = [\langle k, \|u_i\| + \|w_i\|\rangle]_{i=1..m}$ where $k = st - \#\mathcal{K}$. Then $\|\text{pre}(R\ 23)\| > \|\text{post}(R\ 23)\|$ and since the number of submaster constraints remains unchanged, $\|S\|$ decreases.
We show in a similar way as Rule 2 that for Rule 27 $\|S\|$ decreases since $\|X\| > \|u\|$ and therefore $\|X = v\| > \|u = v\|$.
We show in a similar way as Rule 27 that for Rule 29 $\|S\|$ decreases since $\|X\| > \|u\|$ and therefore $\|X \in Forge_c(E', \mathcal{K})\| > \|u \in Forge_c(E', \mathcal{K})\|$.
Rule 30 eliminates one constraint. Then, $\|S\|$ decreases. □
□

8 Termination for Protocols with Autonomous keys

The aim of this section is to prove the termination for our inference system. For this, we need some definitions of index variables.

Definition 34. *(Index Variables)*
$Var_{\mathcal{I}}(t)$ *the set of indexes of* t
$Var_{\mathcal{I}}^{Q}(t)$ *the set of universal indexes of* t
$Var_{\mathcal{I}}^{R}(t)$ *the set of existential indexes of* t
$Var_{\mathcal{I}}^{\mathcal{X},R}(t) = \bigcup_{X_i \leqslant_m t} Var_{\mathcal{I}}^{R}(X_i)$ *where* $X_i \in \mathcal{X}_{\mathcal{I}}$
$Var_{\mathcal{I}}^{\mathcal{C},R}(t) = \bigcup_{c_i \leqslant_m t} Var_{\mathcal{I}}^{R}(c_i)$ *where* $c_i \in \mathcal{C}_{\mathcal{I}}$

Then, we introduce some invariants in Section 8.1 that are used in the proofs of termination. The aim of Section 8.2 is to prove that the set of indexes generated by our inference system is bounded. Once the set of possible indexes to be generated is fixed, we can prove the termination for protocols with autonomous keys by defining a weight for terms, constraints, a constraints block and a system constraint as it is given in Section 8.3.

8.1 Invariants

Invariant 6. *For a constraint* $(t \in Sub(t', E, \mathcal{E}, \mathcal{K}))$ *where* $\exists X_i \leq t'$ *we have* $\forall u \leq t'$ *s.t* $X_i \leqslant_m u$, u *is tagged. For a constraint* $(u = v)$ *where* $\exists X_i < u$, $Y_j < v$ *s.t* $X_i, Y_j \in \mathcal{X}_{\mathcal{I}}$, *we have* X_i *or* Y_j *is tagged. Moreover, if* $X_i \leqslant_m u$ *and* $Y_j \leqslant_m v$ *then* $i = j$.

This invariant will be used in the proof of Invariant 7, Invariant 8 and Proposition 27.

Proof. The first group manages equality constraints that satisfy the invariant. Indeed, Rule 3 eliminates the whole block. Then, the invariant is valid. Rules 6 and 7 transform equality rules and generate new ones.

Nevertheless, these constraints do not fit the shape of constraints defined in Invariant 6, i.e. having two indexed variables in the two terms forming the equality. The rest or rules of the group G_1 does not modify constraints since they only manage labelling. Thus, the invariant is satisfied.
Rules of the group G_2 manage $Forge$ constraints. Only Rule 9 generates a Sub constraint: $(t \in Sub(w, E, \mathcal{E}, \mathcal{K}))$ where $w \in E$. However, according to the second condition of the definition of well-tagged protocols (See Definition 9), w is tagged. Besides, according to the third condition of the same definition, each subterm of w having an indexed variable as a subterm without crossing $mpair$ operator is tagged. Thus, if $\exists X_i \leq t'$ then $\forall u \leq t'$ s.t $X_i \leqslant_m u$, u is tagged, and then, the invariant follows. The other rules do not generate neither equality constraintes, nor Sub ones. Then, the invariant still holds.
The Rule 15 of the group G_3 transforms a Sub constraint into either a Sub_d constraint or an equality one. The Sub_d constraint has the same u, and then, the invariant is still satisfied. The equality constraint is of the form $(t = u)$ where u is the term of the Sub constraint. However, by induction hypothesis, if $\exists X_i \leq u$ then $\forall v \leq u$ s.t $X_i \leqslant_m v$, v is tagged. Then, the first condition of the invariant for the constraint $(t = u)$ is satisfied. Moreover, if $X_i \leqslant_m u$ and $Y_j \leqslant_m t$, then u is tagged since X_i is tagged and according to the third condition of the definition 9. Thus, t has to be tagged with the same tag as u. Thus, $i = j$. The other rules of the group G_3 transform a Sub_d constraint into new Sub constraints while decomposing the term inside the Sub_d constraint. However, the invariant is satisfied for the constraint $(t \in Sub_d(u, E, \mathcal{E}, \mathcal{K}))$ for the term u then it still holds for a subterm u' of u $(u' \leq u)$. Indeed, if $\exists X_i \leq u'$ then, $X_i \leq u$. However, by induction hypothesis, $\forall v \leq u$ s.t $X_i \leqslant_m v$, v is tagged. Therefore, $\forall v \leq u' \leq u$ s.t $X_i \leqslant_m v$, v is tagged. We conclude that all the rules of the group G_3 satisfy the invariant.
Rules of the group G_4 manage equality constraints. Rule 21 eliminates the constraint. Rule 22 eliminates the whole block. Rule 25 replaces an index by another one in all the block. Thus, these rules satisfy the invariant. Rule 23 decomposes an equality constraint of two terms $(u = v)$ into an equality constraint of two subterms $(u' = v')$ where $u' \leq u$ and $v' \leq v$. Thus, if $\exists X_i < u'$, $Y_j < v'$ then $\exists X_i < u$, $Y_j < v$. However, by induction hypothesis, X_i or Y_j is tagged. Moreover, if $X_i \leqslant_m u'$ and $Y_j \leqslant_m v'$ then, $X_i \leqslant_m u$ and $Y_j \leqslant_m v$, and therefore, $i = j$. Rule 24 decomposes an equality constraint of two terms $(mpair(k, u) = mpair(l, w))$ into an equality constraint of two subterms $(u = w\delta_{l,k})$. If $\exists X_i < u < mpair(k, u)$, $Y_j < w\delta_{l,k} < mpair(k, w)$ s.t $X_i, Y_j \in \mathcal{X}_\mathcal{I}$, then, X_i or Y_j is tagged by induction hypothesis. Besides, $Var_\mathcal{I}(u) \subseteq \{k\}$ and $Var_\mathcal{I}(w\delta_{l,k}) \subseteq \{k\}$. Thus, if $X_i \leqslant_m u$ and $Y_j \leqslant_m w\delta_{l,k}$ then $i = k = j$.
Rule 26 of the group G_5 generates an equality constraint $(u = v)$. Suppose $Y_j < u$ and $Z_k < v$. By induction hypothesis, for the constraint $(X_i = u)$, Y_j is tagged. In a similar way, Z_k is tagged. Then, the first condition of the invariant is satisfied. Suppose now that $Y_j \leqslant_m u$ and $Z_k \leqslant_m v$. By induction hypothesis, for the constraint $(X_i = u)$, we have $i = j$. In a similar way, for the constraint $(X_i = v)$, we have $i = k$. Thus, $j = k$. Therefore, the second condition of the invariant is satisfied. For Rule 27, the equality constraint generated $(u = v)$ can not have $X_i \leq u$ or $Y_j \leq v$ by priority of Rules 6 and 7 replacing indexed variables by non indexed ones. In a similar way, Rule 31 can not generate a constraint $(t \in Sub_d(w, E, \mathcal{E}, \mathcal{K}))$ having indexed variables in w. Rules 28, 29 and 30 do not modify equality constraints. Rule 32 eliminates the whole block. Therefore, the invariant is valid for all the rules of the group G_5.
The Rule 33 of the group G_6 generates a constraint $(t \in Sub_d(u\delta, E, \mathcal{E}, \mathcal{K}))$. However, u comes from a master constraint: $(X_i = u)$. By induction hypothesis, if $\exists Y_j < u$ then Y_j is tagged. Moreover, since u is a term of the protocol, according to the third condition of Definition 9, $\forall u' \leq u$ s.t $Y_j \leqslant_m u'$, u' is tagged. Then, the Sub constraint generated satisfies the invariant. Rule 34 does not generate neither Sub constraints nor non final equality ones Then, the invariant still holds. Rule 35 generates an equality constraint $(u_o\delta_o = v)$. Suppose that $\exists Y_j$ and Z_k s.t $Y_j < u_o\delta_o$ and $Z_k < v$. By induction hypothesis for the constraint $(X_m = v)$, we have Z_k is tagged. Then, the first condition of the invariant is satisfied. Suppose now that $Y_j \leqslant_m u_o\delta_o$ and $Z_k \leqslant_m v$. By induction hypothesis for the constraint $(X_m = v)$, we have $m = k$. By induction hypothesis for the constraint $(X_o = u_o)$, where $Y_l < u_o$ we have $o = l$. Then, $Var_\mathcal{I}(u_o\delta_o) \subseteq \{m\}$ which leads to $j = m$. Thus, $j = k$ and therefore the invariant holds for Rule 35. □

Invariant 7. *For a constraint of the form $(t \in Forge_c(E, \mathcal{K}))$, $(t \in Forge(E, \mathcal{K}))$, $(t \in Sub(u, E, \mathcal{E}, \mathcal{K}))$, or $(t \in Sub_d(u, E, \mathcal{E}, \mathcal{K}))$, where $\exists X_i \leqslant_m t$, we have $Var_\mathcal{I}(t) \subset Q$.*
For a non final constraint of the form $u = v$ where $\exists Y_j$ and Z_k s.t $Y_j \leqslant_m u$ and $Z_k \leqslant_m v$, we have $j, k \in Q$.

This invariant will be used in the proof of Invariant 8, Proposition 27, Proposition 28, Lemma 11 and Proposition 31.

Proof. Rules of the group G_1 do not generate new constraints of the form $Forge$ or Sub or equality constraints of the form defined in Invariant 7. Then, the invariant still holds.

Rule 9 of the group G_2 generates from a $Forge$ constraint either a $Forge_c$ constraint or a Sub constraint with the same term t in both cases. However, by induction hypothesis, $Var_{\mathcal{I}}(t) \subset Q$. Then, the invariant is satisfied. Rules 10, 11 and 12 transform a constraint $t \in Forge_c(E, \mathcal{K})$ into another constraint $(t' \in Forge(E, \mathcal{K}))$ where $t' \leqslant_m t$. However, $Var_{\mathcal{I}}(t') \subseteq Var_{\mathcal{I}}(t)$ and $Var_{\mathcal{I}}(t) \subset Q$. Thus, $Var_{\mathcal{I}}(t) \subset Q$ which satisfies the invariant. Rule 13 generates a new constraint $(t \in Forge(E, \mathcal{K}))$ where $Var_{\mathcal{I}}(t) \subset \{k\}$ and $k \in Q$. Thus, the invariant is valid. Rule 14 eliminates the block. Thus, the group G_2 validates the invariant.
Rule 15 transforms a Sub constraint into either a Sub_d constraint preserving the same t or an equality constraint. For the new Sub_d constraint, by induction hypothesis we have $Var_{\mathcal{I}}(t) \subset Q$ which satisfies the invariant. For an equality constraint $(t = u)$, suppose that $\exists X_i, Y_j$ s.t $X_i \leqslant_m t$ and $Y_j \leqslant_m u$. According to Invariant 6 for the constraint $(t = u)$ we have $i = j$. However, by induction hypothesis, $Var_{\mathcal{I}}(t) \subset Q$. Since we only replace an universal index by another universal index, we have $i, j \in Q$. Rules 16 and 19 transform a Sub constraint into another one while preserving the same t. However, by induction hypothesis, $Var_{\mathcal{I}}(t) \subset Q$. Then, the invariant is satisfied. Rules 17 and 18 generate from a Sub constraint a Sub_d constraint with the same t and a new constraint $Forge$ for forging the key b. For the new constraint Sub_d, by induction hypothesis, we have $Var_{\mathcal{I}}(t) \subset Q$ which satisfies the invariant. For the $Forge$ constraint, b is a key. According to the restriction of autonomous keys, we have $Var_{\mathcal{I}}(b) = \emptyset$, which satisfies the invariant. Rule 20 eliminates the block. Thus, the group G_3 satisfies the invariant.
Rule 21 of the group G_4 eliminates a constraint. Rule 22 eliminates the block. Rule 23 transfoms an equality constraint $(u = v)$ into a new one $(u' = v')$ where $u' \leqslant_m u$ et $v' \leqslant_m v$. Then, if $\exists X_i, Y_j$ s.t $X_i \leqslant_m u'$ and $Y_j \leqslant_m v'$, then $X_i \leqslant_m u$ and $Y_j \leqslant_m v$. However, by induction hypothesis, $i, j \in Q$ which satisfies the invariant. Rule 21 transforms an equality constraint into another constraint $(u = w\delta_{l,k})$ where $Var_{\mathcal{I}}(u) = Var_{\mathcal{I}}(v) \subseteq \{k\}$ and $k \in Q$, which satisfies the invariant. Thus, the group G_4 satisfies the invariant.
Rule 26 generates a new constraint $(u = v)$. Suppose that $\exists Z_k, Y_j$ s.t $Z_k \leqslant_m u$ and $Y_j \leqslant_m v$. By induction hypothesis for the constraint $(X_i = u)$, we have $i, k \in Q$. By induction hypothesis for the constraint $(X_i = v)$, we have $i, j \in Q$. Then, $j, k \in Q$, which satisfies the invariant. Rule 27 generates an equality constraint $(u = v)$. However, u and v comes from constraints $(X = u)$ and $(X = v)$. Then, $\nexists X_i$ s.t $X_i \leqslant_m u$ or $X_i \leqslant_m v$ by priority of Rules 6 et 7. In a similar way, Rule 29 does not generate a constraint $(u \in Forge_c(E, \mathcal{K}))$ where $X_i \leqslant_m u$. Thus, Rules 27 and 29 satisfy the invariant. Rule 28 generates a new constraint $(u \in Forge_c(E, \mathcal{K}))$. Suppose that $\exists Y_j$ s.t $Y_j \leqslant_m u$. However, u comes from the constraint $X_i = u$. According to Invariant 6, $i = j$. Moreover, by induction hypothesis, we have $i \in Q$ and therefore $j \in Q$, which satisfies the invariant. Rule 30 does not generate new constraints. Rule 31 generates a new Sub constraint while preserving the same t. Rule 32 eliminates the block. Thus, the group G_5 validates the invariant.
Rule 33 generates a new Sub constraint but preserving the same t. Then, the invariant still holds. Rule 34 generates a new constraint $(u_o\delta_o \in Forge_c(E', \mathcal{K}))$. Suppose that $\exists Y_j$ s.t $Y_j \leqslant_m u_o\delta_o$. However, according to Invariant 6 applied to the master constraint $(X_o = u_o)$ taking into account δ_o, we have $j = m$. Moreover, by induction hypothesis applied to the constraint $(X_m \in Forge_c(E', \mathcal{K}))$, we have $m \in Q$, which satisfies the invariant. Rule 35 generates two new constraints. The first constraint is $(v \in Forge_c(E'_r, \mathcal{K}_r))$. Suppose that $\exists Y_j$ s.t $Y_j \leqslant_m v$. According to the induction hypothesis applied to the constraint $(X_m = v)$, we have $m, j \in Q$, which satisfies the invariant. The second constraint generated is $(u_o\delta_o = v)$. Suppose that $\exists Y_j, Z_k$ s.t $Y_j \leqslant_m v$ and $Z_k \leqslant_m u_o\delta_o$. However, by induction hypothesis applied to both the master constraint $(X_o = u_o)$ taking into account δ_o and the constraint $(X_m = v)$, we have $m, j, k \in Q$ which satisfies the invariant. Thus, the group G_6 validates the invariant. □

Invariant 8. *For a constraint $(u = v)$, where $X_i \leqslant_m u$ and $c_j \leqslant_m v$, then i and j can not be both existential indexes.*

Proof. Groups G_1 and G_2 do not generate new equality constraints of the form $(u = v)$ where $\exists X_i \in \mathcal{X}_{\mathcal{I}}$ and $c_j \in \mathcal{C}_{\mathcal{I}}$ s.t $X_i \leqslant_m u$ and $c_j \leqslant_m v$. Thus, the invariant still holds.
For the group G_3, only Rule 15 can generate an equality constraint $(t = u)$ from a constraint $(t \in Sub(u, E, \mathcal{E}, \mathcal{K}))$. Suppose that $\exists X_i \in \mathcal{X}_{\mathcal{I}}$ and $c_j \in \mathcal{C}_{\mathcal{I}}$ s.t $i, j \in R$ and either $(X_i \leqslant_m u$ and $c_j \leqslant_m t)$ or $(X_i \leqslant_m t$ and $c_j \leqslant_m u)$. In the first case, i.e. $(X_i \leqslant_m u$ and $c_j \leqslant_m t)$, according to Invariant 6, X_i is tagged. According to the third condition of Definition 9, u is also tagged. Thus, v has to be tagged with the same tag as u and then $i = j$, which contradicts our hypothesis. In the second case, i.e. $X_i \leqslant_m t$ and $c_j \leqslant_m u$, according to the Invariant 7, we have $i \in Q$ which contradicts the hypothesis: $i \in R$. Thus, for the two cases, i and j can not be both existentials.
Rule 21 eliminates a constraint. Rule 22 eliminates the whole block. Rule 25 replaces an index by another one. Nevertheless, it can not replace an universal index by an existential one. Thus, these three rules of the group G_4 satisfy the invariant. Rule 23 decomposes an equality constraint between two terms $(u = v)$ into another

equality contraint between two subterms ($(u' = v')$ with $u' <_m u$ and $v' <_m v$) without generating indexes. By induction hypothesis, if $\exists X_i \in \mathcal{X}_\mathcal{I}$ and $c_j \in \mathcal{C}_\mathcal{I}$ s.t $X_i \leqslant_m u' <_m u$ and $c_j \leqslant_m v' <_m v$ then i and j can not be both existential. Rule 24 generates new equality constraint $(u = w\delta_{l,k})$ with $Var_\mathcal{I}(u) = Var_\mathcal{I}(w\delta_{l,k}) \subseteq \{k\}$ which satisfies the invariant.
Rule 26 of the group G_5 generates an equality constraint $(u = v)$. Suppose that $\exists Z_k \in \mathcal{X}_\mathcal{I}$ and $c_j \in \mathcal{C}_\mathcal{I}$ s.t $k, j \in R$, $Z_k \leqslant_m u$ and $c_j \leqslant_m v$. Nevertheless, according the invariant 6 applied to the constraint $(X_i = u)$, we have $i = k$. However, according to the induction hypothesis for the constraint $(X_i = v)$, i and j can not be both existential. Thus, k and j can not be both existential which satisfies the invariant. Rule 27 generates an equality constraint $(u = v)$. However, u and v come from constraints $(X = u)$ and $(X = v)$. The, $\nexists X_i$ s.t $X_i \leqslant_m u$ or $X_i \leqslant_m v$ by priority of Rules 6 and 7. Thus, Rule 27 satisfies the invariant. The other rules of the group G_5 do not generate new equality constraints. Thus, the group G_5 satisfies the Invariant 8.
Rules 33 and 34 of the group G_6 do not generate new not final equality constraints. Thus, the invariant is satisfied. Rule 35 generates a new equality constraint $(u_o\delta_o = v)$. Suppose that $\exists Z_k \in \mathcal{X}_\mathcal{I}$ and $c_j \in \mathcal{C}_\mathcal{I}$ s.t and either $Z_k \leqslant_m u_o\delta_o$ and $c_j \leqslant_m v$, or $Z_k \leqslant_m v$ and $c_j \leqslant_m u_o\delta_o$. In the first case, i.e. $Z_k \leqslant_m u_o\delta_o$ and $c_j \leqslant_m v$, according to Invariant 6 applied to the master constraint $(X_o = u_o)$ taking into account δ_o, we have $k = m$. Moreover, by induction hypothesis for the constraint $(X_m = v)$, m and j can not be both existential. Then, k and j can not be both existential, which satisfies the invariant. In the second case, i.e. $Z_k \leqslant_m v$ and $c_j \leqslant_m u_o\delta_o$, according to the invariant 6 for the constraint $(X_m = v)$, we have $m = k$. However, by induction hypothesis for the master constraint $(X_o = u_o)$ taking into account δ_o, we have m and j can not be both existential and then k and j can not be both existential, which satisfies the invariant. Thus, in both cases, the invariant is valid. □

Invariant 9. *For a constraint $t \in Sub(u, E, \mathcal{E}, \mathcal{K})$ or $t \in Sub_d(u, E, \mathcal{E}, \mathcal{K})$ where $Var^R_\mathcal{I}(u) = \{i\}$ we have either i is fresh and $(ant(i, j) = j$ if $Var^{\mathcal{C},R}_\mathcal{I}(t) = j \in R)$ or $i \in Var^R_\mathcal{I}(\mathcal{E})$.*

This invariant is used for the proof of Invariant 10, Proposition 27 and Proposition 28.

Proof. we show that the different rules of our inference sytem satisfy the invariant. Rules of the first group does not generate Sub constraints. Then, the invariant holds. Rule 9 generates a constraint $(t \in Sub(w, E, \mathcal{E}, \mathcal{K}))$ where $w \in E$. and then, $Var_\mathcal{I}(w) = \emptyset$. The other rules of the second group do not generate Sub constraints. Thus, the invariant is satisfied. Rule 15 transforms a Sub constraint into a Sub_d constraint while preserving the same u. Then, the invariant still holds. Rules 16, 17, 18 transform a constraint $(t \in Sub_d(u, E, \mathcal{E}, \mathcal{K}))$ into one single constraint $(t \in Sub(u', E, \mathcal{E}, \mathcal{K}))$ where $u' \leqslant_m u$. If $Var^R_\mathcal{I}(u') = \{i\}$ then, $Var^R_\mathcal{I}(u) = \{i\}$. However, by induction hypothesis, either i is fresh and $(ant(i, j) = j$ if $Var^{\mathcal{C},R}_\mathcal{I}(t) = j \in R)$ or $i \in Var^R_\mathcal{I}(\mathcal{E})$ which satisfies the invariant. For Rule 19, there are two cases. In the first one, it generates a new Sub constraint with fresh index variable k and thus if $Var^{\mathcal{C},R}_\mathcal{I}(t) = j \in R$ then $ant(i, j) = j$. Therefore, the invariant is satisfied. In the second case, it generates an old constraint that exists in $Hist(B)$. Then, by induction hypothesis, the invariant is satisfied. Rule 20 eliminates the whole block. We conclude that the third group satisfies the invariant. Rules of the fourth group do not generate Sub constraints. Only Rule 25 may generate a new Sub constraint with a new existential index inside the Sub. In this case we have $(t \in Sub(u, E, \mathcal{E}, \mathcal{K}))$ or $(t \in Sub_d(u, E, \mathcal{E}, \mathcal{K}))$ where $Var^R_\mathcal{I}(u) = \{i\}$. We must have also in the block $(c_i = c_k)$ where $i, k \in R$ in order to have a constraint of the form $(t \in Sub(u', E, \mathcal{E}, \mathcal{K}))$ or $(t \in Sub_d(u', E, \mathcal{E}, \mathcal{K}))$ where $Var^R_\mathcal{I}(u') = \{k\}$. However, by induction hypothesis, either i is fresh and $(ant(i, j) = j$ if $Var^{\mathcal{C},R}_\mathcal{I}(t) = j \in R)$ or $i \in Var^R_\mathcal{I}(\mathcal{E})$. If i is fresh, then it is contradictory with $(c_i = c_k)$ in the block. If $i \in Var^R_\mathcal{I}(\mathcal{E})$, then $ant(i, j) = \{k\}$ if $k \in \mathcal{E}$ and $ant(i, k) = \{i\}$ otherwise. In these two cases, we have $Var^R_\mathcal{I}(u') \in Var^R_\mathcal{I}(\mathcal{E})$ which satisfies the invariant. Only Rule 31 generatse a Sub_d constraint: $(t \in Sub_d(w, E, \mathcal{E}, \mathcal{K}))$. However, w comes from a sub master constraint. Then, $Var^R_\mathcal{I}(w) \in Var^R_\mathcal{I}(\mathcal{E})$ which satisfies the invariant. Only Rule 33 generate a Sub_d constraint. There are two cases. In the first one, it generates a new Sub_d constraint with fresh index variable k' and then $(ant(i, j) = j$ if $Var^{\mathcal{C},R}_\mathcal{I}(t) = j \in R)$. Thus, the invariant is satisfied. In the second case, it generates an old constraint that already exists in $Hist(B)$. Then, by induction hypothesis, the invariant is satisfied. □

8.2 Bound on Indexes Generated by the Inference System

The aim of this section is to bound the set of possible indexes generated by our inference system. For this, we consider a derivation i.e a successive application of our rules on a constraint system. In this derivation, the number of application of labelling rules and modification of sub-master constraints, i.e. Rules 4, 5 and 8 is bounded. We consider a derivation where there is not modification or labelling of sub-master constraints ($\mathcal{E}$ is

then fixed for all rules). Indexes in $\mathcal{E}$ are the most anterior indexes. The aim of this section is then to bound the set of indexes generated by our inference system throught this derivation.

Theorem 3. *The set of indexes generated by our inference system is bounded.*

Proof. We first define this set of indexes useful for the proof:

Definition 35. *Let:*

$V_1 = Var_{\mathcal{I}}^{\mathcal{C},R}(\{t | t \in Forge(E,\mathcal{K}) \vee t \in Forge_c(E,\mathcal{K}) \vee t \in Sub(u,E,\mathcal{E},\mathcal{K}) \vee t \in Sub_d(u,E,\mathcal{E},\mathcal{K}) \in B\})$

$V_2 = Var_{\mathcal{I}}^{R}(\{u | t \in Sub(u,E,\mathcal{E},\mathcal{K}) \vee t \in Sub_d(u,E,\mathcal{E},\mathcal{K}) \in B \text{ and } \exists X_i \text{ s.t } X_i \leqslant_m t\})$

$V_3 = Var_{\mathcal{I}}^{\mathcal{C},R}(\{v | u = v \vee v = u \in B \text{ and } \exists X \text{ s.t } X \leqslant_m u \text{ and } X \in \mathcal{X} \cup \mathcal{X}_{\mathcal{I}}\})$

$V_4 = Var_{\mathcal{I}}^{R}(\mathcal{E})$

Then, $V = V_1 \cup V_2 \cup V_3 \cup V_4$.

Then, the proof of Theorem 3 is structured as follows:
We bound first the set of universal indexes generated by our inference system (See Proposition 26).
Then, we bound the set of existential indexes generated by our inference system. First, considering a constraint $(t \in Sub(v,E,\mathcal{E},\mathcal{K}))$, while fixing t (by the condition $X_i \leqslant_m t$), we bound the set of existential indexes generated by our inference system inside a Sub (in v). For this, we bound first the set of existential indexes indexing variables inside a Sub (See Proposition 27). Once this set is bounded, we bound the set of existential indexes indexing constants inside a Sub (See Proposition 28) permetting then to bound the set of existential indexes inside a Sub. Then, we bound the set of existential indexes in V defined in Definition 35 (See Proposition 29). We can after bound the set of existential indexes in u for constraints of the form $(t \in Sub(u,E,\mathcal{E},\mathcal{K}))$ without any restrictions over the term t (See Proposition 30). Once the of existential indexes for $Forge$ and Sub constraints is bounded, we bound existential indexes for equality constraints (See Proposition 31 and Proposition 32).

Thus, the proof of Theorem 3 follows from the following propositions:

Proposition 26. *The set of universal indexes generated by our inference system is bounded.*

Proposition 27. $Var_{\mathcal{I}}^{\mathcal{X},R}(\{v \text{ s.t } (t \in Sub(v,E,\mathcal{E},\mathcal{K})) \text{ or } (t \in Sub_d(v,E,\mathcal{E},\mathcal{K})) \in B \text{ and} \exists X_i \leqslant_m t\}) \cup Var_{\mathcal{I}}^{R}(\mathcal{E})$ *is bounded.*

Proposition 28. $Var_{\mathcal{I}}^{\mathcal{C},R}(\{v \,|\, (t \in Sub(v,E,\mathcal{E},\mathcal{K})) \vee (t \in Sub_d(v,E,\mathcal{E},\mathcal{K})) \in B \text{ and } \exists X_i \leqslant_m t)\}$ *has a bound.*

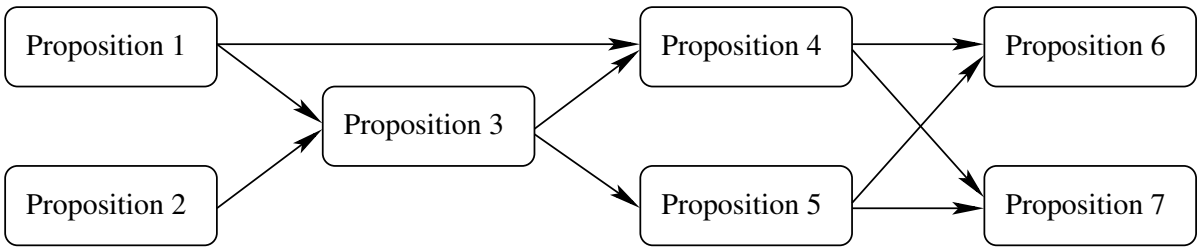
Proposition 29. *The set of indexes in* V *(defined in Definition 35) is bounded.*

Proposition 30. $Var_{\mathcal{I}}^{R}(\{u \,|\, t \in Sub(u,E,\mathcal{E},\mathcal{K}) \vee t \in Sub_d(u,E,\mathcal{E},\mathcal{K}) \in B\})$ *is bounded.*

Proposition 31. $Var_{\mathcal{I}}^{\mathcal{X},R}(\{u \,|\, (v = u) \vee (u = v) \in B\})$ *is bounded.*

Proposition 32. $Var_{\mathcal{I}}^{\mathcal{C},R}(\{u \,|\, (v = u) \vee (u = v) \in B\})$ *is bounded.*

The dependency between these propositions is illustrated by Figure8.2.



□

Before proving the propositions above, we need to prove the following invariant:

Invariant 10. *For a non final or dull constraint* $(u = v)$ *where* $Var_{\mathcal{I}}^{\mathcal{C},R}(u) = \{i\}$ *and* $Var_{\mathcal{I}}^{\mathcal{C},R}(v) = \{j\}$*, we have* $ant(i,j) \in V$ *where* V *is defined in Definition 35.*

This invariant will be used in Invariant 11, Proposition 31 and Proposition 32.

Proof. Rules of the first group and the second group do not generate constraints of the form defined in Invariant 10. Then, the Invariant is satisfied.

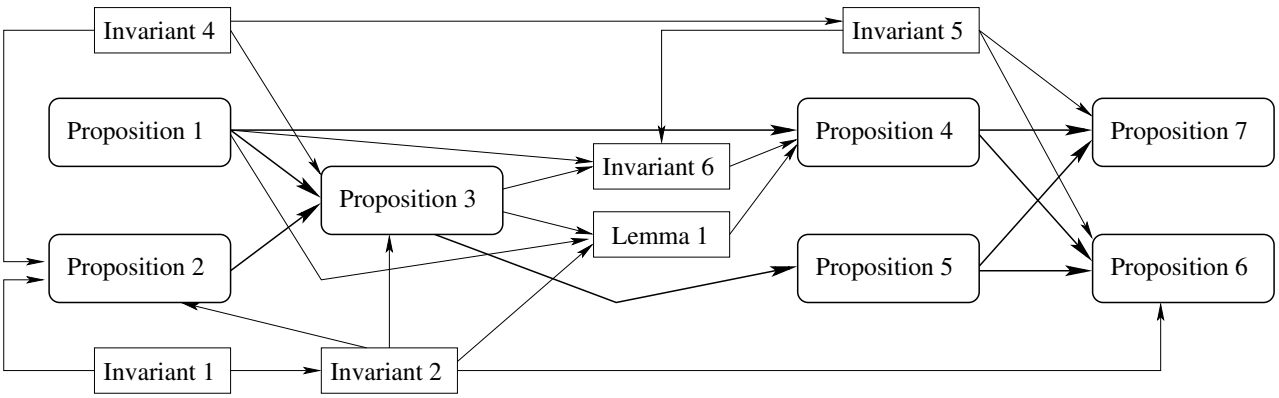
Only Rule 15 generate an equality constraint $(u = v)$ from a constraint $(u \in Sub(v, E, \mathcal{E}, \mathcal{K}))$ where $Var_{\mathcal{I}}^{\mathcal{C},R}(u) = \{i\}$ and $Var_{\mathcal{I}}^{\mathcal{C},R}(v) = \{j\}$. However, according to Invariant 9, either j is fresh and $ant(i,j) = i$ or $j \in Var_{\mathcal{I}}^{R}(\mathcal{E})$. There are then two cases. In the first case, j is fresh and $ant(i,j) = i$. However, $i \in V$ since $i \in V_1$. Thus, $ant(i,j) \in V$ and the invariant is satisfied. In the second case, $j \in Var_{\mathcal{I}}^{R}(\mathcal{E})$. Then, $ant(i,j) = i$ if $i \in Var_{\mathcal{I}}^{R}(\mathcal{E})$ and $ant(i,j) = j$ otherwise. In both cases, $ant(i,j) \in V$ and the invariant follows. The same reasoning is valid when the constraint $(v \in Sub(u, E, \mathcal{E}, \mathcal{K}))$ is transformed into a constraint $(u = v)$.

Rule 21 eliminates the constraint. Rule 22 eliminates the whole block. Rule 23 decomposes an equality between two terms into an equality between two subterms while preserving the same indexes. Then, th invariant still holds. Rule 24 generates an equality constraint $(u = v)$ where $Var_{\mathcal{I}}(u) = Var_{\mathcal{I}}(v)$. Then, the invariant remains satisfied. Rule 25 replace an index by another in the whole block. It may lead to a generation of a new constraint $(u = v')$ from a constraint $(u = v)$ where $Var_{\mathcal{I}}(v) = \{j\}$, $Var_{\mathcal{I}}(v') = \{k\}$, $Var_{\mathcal{I}}(u) = \{i\}$ and $i, j, k \in R$. We must then have the constraint $(c_j = c_k)$ in the block. Then, we have $ant(j,k) = k$. We distinguish two cases concerning the constraint $(u = v')$. In the first one, $ant(i,k) = k$. However, by induction hypothesis for the constraint $(c_j = c_k)$ and since $ant(j,k) = k$, we have $k \in V$ which satisfies the invariant. In the second case, $ant(i,k) = i$. However, $ant(j,k) = k$. Then $ant(i,j) = i$ and therefore by induction hypothesis for the constraint $(u = v)$ we have $i \in V$ which satisfies the invariant. Thus, in both cases, the invariant still holds.

Only Rules 26 and 27 of the fifth group may generate new constraints of the form $(u = v)$. However, u and v belongs to constraints of the form $(X_i = u)$ or the form $(X = u)$ whose existential indexes are already counted in V. Thus,, Group G_5 satisfies the invariant.

Only Rule 35 may generate a new equality constraint of the form $(u_o\delta_o = v)$. Suppose $Var_{\mathcal{I}}^{\mathcal{C},R}(u_o\delta_o) = \{i\}$ and $Var_{\mathcal{I}}^{\mathcal{C},R}(v) = \{j\}$. There are two cases. In the first one, it generates a new equality constraint $(u_o\delta_o = v)$ with fresh index variable i and then $ant(i,j) = j$. However, $j \in V_3$. Then, $ant(i,j) \in V$. Thus, the invariant is satisfied. In the second case, it generates an old constraint that already exists in $Hist(B)$. Then, by induction hypothesis, the invariant is satisfied. □

Figure 8.2 details the dependency between the different propositions and invariants.



Proposition 26. The set of universal indexes generated by our inference system is bounded.

This proposition will be used in the proof of Theorem 3, Proposition 28 and Lemma 11.

Proof. There are only two rules that may generate universal indexes in constraints, i.e. Rules 13 and 24. Then, the number of universal indexes is bounded by the number of constraints of the form $(mpair(i,t) \in Forge_c(E, \mathcal{K}))$ and $(mpair(j,u) = mpair(k,v))$.

The number of constraints of the form $(mpair(i,t) \in Forge_c(E, \mathcal{K}))$ is bounded by (the number of $mpair$s $*$ the number of different $E *$ the number of different $\mathcal{K}$). Let st the number of subterms of the protocol. The

number of *mpair*s is bounded by *st*. The number of different E is bounded by 2^{st} since terms in E have not indexes outside *mpair*s according to the restriction of Well-Tagged protocols (See Definition 9). The number of different $\mathcal{K}$ is bounded by 2^{keys} which is bounded by 2^{st} since keys have not indexes outside *mpair*s according to the restriction of autonmous keys (See Definition 10).

The number of constraints of the form $(mpair(j,u) = mpair(k,v))$ is bounded by (the number of *mpair*s $*$ the number of *mpair*s) which is bounded by $(st * st)$. We conclude that the number of universal indexes is bounded. □

Proposition 27. $Var_{\mathcal{I}}^{\mathcal{X},R}(\{v \text{ s.t } (t \in Sub(v,E,\mathcal{E},\mathcal{K})) \text{ or } (t \in Sub_d(v,E,\mathcal{E},\mathcal{K})) \in B \text{ and } \exists X_i \leqslant_m t\}) \cup Var_{\mathcal{I}}^{R}(\mathcal{E})$ is bounded.

This proposition will be used for the proof of Proposition 28 and Theorem 3.

Proof. Rule 19 generates a bounded set of indexes. Indeed, let *st* the number of subterms of the protocol. Then, the number of existential indexes in v from constraints $(t \in Sub(v,E,\mathcal{E},\mathcal{K}))$ or $(t \in Sub_d(v,E,\mathcal{E},\mathcal{K}))$ is bounded by the number of constraints of this form which is bounded by: (the number of possible terms t $*$ the number of different E $*$ the number of different $\mathcal{K}$) $*$ the number of existential indexes in v fixing the other parameters. However, $\exists X_i \leqslant_m t$ and according to Invariant 7, $i \in Q$. Then, the number of indexes in t is bounded. Let b_1 this bound. Thus, the number of possible terms t is bounded by $2^{b_1 * st}$. Besides, the number of different E is bounded by 2^{st} since terms in E have not indexes outside *mpair*s according to the restriction of Well-Tagged protocols (See Definition 9). Moreover, the number of different $\mathcal{K}$ is bounded by 2^{keys} which is bounded by 2^{st} since keys have not indexes outside *mpair*s according to the restriction of autonmous keys (See Definition 10). Then, fixing t, E, $\mathcal{K}$ and $\mathcal{E}$ (which is fixed by the hypothesis on the derivation chosen), the number of new existential indexes in v generating by Rule 19 is the number of *mpair*s which is bounded by *st*.

We now prove Proposition 27 by proving the following invariant:
$Var_{\mathcal{I}}^{\mathcal{X},R}(\{v \text{ s.t } (t \in Sub(v,E,\mathcal{E},\mathcal{K})) \text{ or } (t \in Sub_d(v,E,\mathcal{E},\mathcal{K})) \in B \text{ and } \exists X_i \leqslant_m t\}) \cup Var_{\mathcal{I}}^{R}(\mathcal{E})$ is stable or decreases by our inference system except Rule 19.
We denote by V the set of indexes defined in Proposition 27. We recall that in the derivation that we consider, $\mathcal{E}$ is fixed. Then, $Var_{\mathcal{I}}^{R}(\mathcal{E})$ is fixed. Rules of the group G_1 does not generate *Sub* constraints. Then, the invariant is valid.
Rule 9 of the group G_2 generates a constraint $(t \in Sub(w,E,\mathcal{E},\mathcal{K}))$ with $w \in E$. Then, $Var_{\mathcal{I}}(w) = \emptyset$, which satisfies the invariant. The other rules of the group G_2 do not generate *Sub* constraints. Thus, the group G_2 satisfies the invariant.
Rule 15 transforms a *Sub* constraint into either an other Sub_d constraint while preserving the same u and then V is stable or an equality constraint and then V may decrease. Rules 16, 17 and 18 transform a Sub_d constraint into a *Sub* constraint while decomposing the term inside a *Sub* without generating new indexes. Then, V remains stable or decreases. Rule 20 eliminates the whole block, then V remains stable or decreases. Rules of the group G_4 do not generate new *Sub* constraints. Besides, Rule 25 does not replace an universel index by an existential one. The only case where Rule 25 may generate new index inside a *Sub* is to transform a constraint $(t \in Sub(v,E,\mathcal{E},\mathcal{K}))$ or $(t \in Sub_d(v,E,\mathcal{E},\mathcal{K}))$ where $Var_{\mathcal{I}}(v) = \{i\}$ and $i \in R$ into a constraint $(t \in Sub(v',E,\mathcal{E},\mathcal{K}))$ or $(t \in Sub_d(v',E,\mathcal{E},\mathcal{K}))$ where $Var_{\mathcal{I}}(v') = \{j\}$ and $j \in R$ by a replacement thanks to the application of Rule 25 to the constraint $(c_i = c_j)$. However, according to Invariant 9, i is fresh or $i \in Var_{\mathcal{I}}^{R}(\mathcal{E})$. The first case ($i$ is fresh) is contradictory with the existence of $(c_i = c_j)$. In the second case $(i \in Var_{\mathcal{I}}^{R}(\mathcal{E}))$, the only way to replace i by j is to have $ant(i,j) = \{j\}$. Since $i \in Var_{\mathcal{I}}^{R}(\mathcal{E})$, this case is only possible if $j \in Var_{\mathcal{I}}^{R}(\mathcal{E})$. Thus, V remains stable or decreases.
Rule 31 generates a constraint $(t \in Sub_d(w,E,\mathcal{E},\mathcal{K}))$. Suppose that $\exists X_i$ s.t $X_i \leqslant_m w$ and $i \in R$. However, this is contradictary with the fact that X_i comes from the constraint $(X = w)$ by priority of Rules 6 and 7. The other rules of the group G_5 do not generate *Sub* constraints. Then, V remains stable.
Rule 33 generates a constraint $(t \in Sub_d(u\delta,E,\mathcal{E},\mathcal{K}))$. Suppose that $\exists Y_j$ s.t $Y_j \leqslant_m u\delta$ and $j \in R$. However, according to Invariant 6 for the master constraint taking into account δ, we have $j = m$ and then V remains stable. The other rules of the group G_6 do not generate new *Sub* constraints. Thus, V remains stable. □

Proposition 28. $Var_{\mathcal{I}}^{\mathcal{C},R}(\{v \mid (t \in Sub(v,E,\mathcal{E},\mathcal{K})) \vee (t \in Sub_d(v,E,\mathcal{E},\mathcal{K})) \in B \text{ and } \exists X_i \leqslant_m t)\}$ has a bound.

This proposition will be used for the proof of Theorem 3, Lemma 11, Invariant 11 and Proposition 30.

Proof. We recall that we consider a derivation where we have $\mathcal{E}$ fixed. Then, let b_2 be the number of constraints of $\mathcal{E}$ and $id = Var^R_{\mathcal{I}}(\mathcal{E})$. Then, let st the number of subterms of the protocol.

The number of existential indexes in v from constraints $(t \in Sub(v, E, \mathcal{E}, \mathcal{K}))$ or $(t \in Sub_d(v, E, \mathcal{E}, \mathcal{K}))$ is bounded by the number of constraints of this form which is bounded by: (the number of possible terms t $*$ the number of different E $*$ the number of different $\mathcal{K}$) $*$ the number of existential indexes in v fixing the other parameters.

However, $\exists X_i \leqslant_m t$ and according to Invariant 7, $i \in Q$. Then, the number of indexes in t is bounded. Let b_1 this bound. Thus, the number of possible terms t is bounded by $2^{b_1 * st}$. Besides, the number of different E is bounded by 2^{st} since terms in E have not indexes outside *mpair*s according to the restriction of Well-Tagged protocols (See Definition 9). Moreover, the number of different $\mathcal{K}$ is bounded by 2^{keys} which is bounded by 2^{st} since keys have not indexes outside *mpair*s according to the restriction of autonmous keys (See Definition 10).

Fixing t, E, $\mathcal{E}$ and $\mathcal{K}$ for the constraint $(t \in Sub(v, E, \mathcal{E}, \mathcal{K}))$ or $(t \in Sub_d(v, E, \mathcal{E}, \mathcal{K}))$ there are four possible rules that may generate existential indexes in v, i.e. Rules 19, 31, 33 and 25. Then, we distinguish four cases depending on the rule that leads to the generation of an existential index.

In the first case, i.e. Rule 19 is the rule that generates a new existential index. The number of possible existential indexes in v is the number of *mpair*s which is bounded by st.

In the second case, i.e. Rule 31 is the rule that generates a new existential index. Then, the number of possible existential indexes in v is the number of constraints of the form $(X = v)^{sm}$ which is bounded by b_2.

In the third case, i.e. Rule 33 is the rule that generates a new existential index. Since we generate only one existential index by indexed variable and by master constraint for this variable, the number of possible existential indexes in v is the number possible replacement which is bounded by (the number of indexed variables $*$ the number of master constraints). The number of master constraints is bounded by b_2. The number of indexed variables is bounded by (the number of vectors $*$ the number of indexes inside the Sub for indexed variables). The number of vectors is fixed from the specification of the protocol. The number of indexes inside the constraint Sub is the sum of the number of universal indexes which is bounded in Proposition 26 and the number of existential indexes (for indexed variables) which is bounded in Proposition 27. Thus, the number of existential indexes in v generated by Rule 33 is bounded.

In the fourth case, i.e. Rule 25 is the rule that generates a new existential index inside the Sub. This is only possible when it transforms a constraint $(t \in Sub(v, E, \mathcal{E}, \mathcal{K}))$ or $(t \in Sub_d(v, E, \mathcal{E}, \mathcal{K}))$ where $Var_{\mathcal{I}}(v) = \{i\}$ and $i \in R$ into a constraint $(t \in Sub(v', E, \mathcal{E}, \mathcal{K}))$ or $(t \in Sub_d(v', E, \mathcal{E}, \mathcal{K}))$ where $Var_{\mathcal{I}}(v') = \{j\}$ and $j \in R$ by a replacement thanks to the application of Rule 25 to the constraint $(c_i = c_j)$.

However, according to Invariant 9, i is fresh or $i \in Var^R_{\mathcal{I}}(\mathcal{E})$. The first case ($i$ is fresh) is contradictory with the existence of $(c_i = c_j)$. In the second case $(i \in Var^R_{\mathcal{I}}(\mathcal{E}))$, the only way to replace i by j is to have $ant(i, j) = \{j\}$. Since $i \in Var^R_{\mathcal{I}}(\mathcal{E})$, this case is only possible if $j \in Var^R_{\mathcal{I}}(\mathcal{E})$. Then, the number of indexes generated inside the Sub in this case is bounded by id. □

Proposition 29. The set of indexes in V (defined in Definition 35) is bounded.

This proposition will be used for the proof of Proposition 31 and Proposition 32.

Proof. The proof is a direct consequence of the following Invariant 11 and the following Lemma 11.

Lemma 11. *The set of indexes in V (defined in Definition 35) generated by the following rules:*
Rule 19 for a constraint $(t \in Sub_d(u, E, \mathcal{E}, \mathcal{K}))$ when $\exists Z_j \leqslant_m t$;

Rule 33 for a constraint $(t \in Sub_d(X_m, E, \mathcal{E}, \mathcal{K}))$ when $\exists Z_j \leqslant_m t$;

Rule 34;

Rule 35 for a constraint $(X_m = v)$ when $\exists Z_m \leqslant_m v$.

is bounded.

Proof. Rule 19 generates a new constraint $t \in Sub(u\delta, E, \mathcal{E}, \mathcal{K})$ that may contain a new existential index in $u\delta$. However, since $\exists Z_j \leqslant_m t$ and according to Proposition 28 and Proposition 27, the set of existential indexes that may be generated by Rule 19 is bounded. In the same way, Rule 33 may generate a new existential index inside the constraint Sub, but that are bounded according to both Proposition 28 and Proposition 27.
Rule 34 may generate new existential indexes in the constraint $u_o\delta_o \in Forge(E', \mathcal{K})$. However the set of these

indexes is bounded. Indeed, the number of existential indexes in $u_o\delta_o$ is bounded by the number of constraints of the form $u_o\delta_o \in Forge(E', \mathcal{K})$ which is bounded by: (the number of different K $*$ the number of different E' $*$ the number of existential indexes in $u_o\delta_o$ fixing E' and $\mathcal{K}$).

However, the number of different E is bounded by 2^{st} since terms in E have not indexes outside *mpair*s according to the restriction of Well-Tagged protocols (See Definition 9). Besides, the number of different $\mathcal{K}$ is bounded by 2^{keys} which is bounded by 2^{st} since keys have not indexes outside *mpair*s according to the restriction of autonmous keys (See Definition 10).

Moreover, the number of existential indexes in $u_o\delta_o$ fixing E' and $\mathcal{K}$ is the number of possible replacements which is bounded by (the number of indexed variable in $Forge$ constraints $*$ the number of master constraints) since we have exactly one fresh index per indexed variable per master constraint. The number of master constraints is bounded by the number of constraints in $\mathcal{E}$ which is fixed. The number of indexed variables in $Forge$ constraints is bounded by (the number of vectors $*$ the number of indexes in variables for $Forge$ constraints). However, according to Invariant 7, these indexes are universal. Then, according to Proposition 26, the set of universal indexes is bounded. Besides, the number of vectors is fixed from the specification of the protocol.

For Rule 35, considering a constraint $(X_m = v)$ and since $\exists Z_m \leqslant_m v$, we have $m \in Q$ according to Invariant 7. Then, for this rule, we have existential indexes only in $u_o\delta_o$. Therefore, the number of fresh indexes generated by this rule is bounded by the number of constraints of the form $(u_o\delta_o = v)$ which is bounded by ($(st$ $*$ the set of universal indexes) $*$ the set of existential indexes in $u_o\delta_o$ fixing v). However, the set of universal indexes is bounded according to Proposition 26. Besides, the set of existential indexes in $u_o\delta_o$ fixing v is bounded by (the number of indexed variables in equality constraints $*$ the number of master constraints) since we have exactly one fresh index per indexed variable in equality constraint per master constraint. However, the number of master constraints is bounded by the number of constraints in $\mathcal{E}$ which is fixed. Moreover, the number of indexed variables in equality constraints of the form $(X_m = v)$ is bounded by (the number of vectors $*$ the number of indexes in variables for these equality constraints). Since $m \in Q$ and the number of vectors is fixed and according to Proposition 26, this number of indexed variables is bounded.

We conlude that the set of indexes in V generated by these rules is bounded. □

Invariant 11. *V is stable or decreases by application of our inference system except:*
Rule 19 for a constraint $(t \in Sub_d(u, E, \mathcal{E}, \mathcal{K}))$ when $\exists Z_j \leqslant_m t$;

Rule 33 for a constraint $(t \in Sub_d(X_m, E, \mathcal{E}, \mathcal{K}))$ when $\exists Z_j \leqslant_m t$;

Rule 34;

Rule 35 for a constraint $(X_m = v)$ when $\exists Z_m \leqslant_m v$.

Proof. V_4 is always stable by hypothesis on the chosen derivation. V_2 is bounded according to Proposition 27 and Proposition 28. Then, in this proof, we consider this bound and show that, for each rule of our inference system, the possible indexes that we may generate in V_1 and V_3 exist always either in V_2 and V_4 that are fixed or in V_1 and V_3 before the application of the rule. Rules of the group G_1 do not generate new $Forge$ or Sub constraints, then V_1 remains stable. Moreover, rules of this group do not generate new equality constraints $u = v$ with $\exists X_i, Y_j \in \mathcal{X}_\mathcal{I}$ s.t $X_i \leqslant_m u$ and $Y_j \leqslant_m v$. Thus, V remains stable or decreases by the group G_1.

Rules of the group G_2 do not generate equality constraints, then V_3 remains stable. Rule 9 transforms a $Forge$ constraint either into a $Forge$ constraint or a Sub constraint with the same t in both cases. Then, V_1 remains stable. Rule 13 generates a new constraint $(t \in Forge(E, \mathcal{K}))$ with $Var_\mathcal{I}(t) \subseteq \{k\}$ and $k \in Q$. Then, V_1 remains stable. Rule 14 eliminates the whole block. The other rules of the group G_2 transform a constraint $(t \in Forge_c(E, \mathcal{K}))$ into a constraint $(t' \in Forge(E, \mathcal{K}))$ with $mpair(t', t)$ without generating new indexes. Then, V_1 remains stable. Thus, V is stable by the group G_2.

Rule 15 transforms a constraint $(t \in Sub(u, E, \mathcal{E}, \mathcal{K}))$ into either another Sub_d constraint with the same t and then V_1 remains stable or an equality constraint $(t = u)$. Suppose that $\exists X_i$ s.t $X_i \leqslant_m t$ or $X_i \leqslant_m u$. We distinguish two cases. In the first one, $X_i \leqslant_m t$. Suppose that $\exists c_j$ s.t $c_j \leqslant_m u$ and $j \in R$. However, j already belongs to V_2. Thus, $V_2 \cup V_3$ remains stable and then V is stable. In the second case, $X_i \leqslant_m u$. Suppose that $\exists c_j$ s.t $c_j \leqslant_m t$ and $j \in R$. However, j already belongs to V_1. Thus, $V_1 \cup V_3$ remains stable and then V is stable.

Rule 16 transforms a Sub_d constraint into another Sub constraint while preserving the same t but decomposing the term inside the Sub without generating new indexes. Then, V_1 remains stable. There are not new equality constraint. Thus, V_3 remains stable and then V remains stable. Rule 17 transforms a Sub_d constraint into a

Sub conctraint preserving the same t and generating a $Forge$ constraint. The last constraint (the $Forge$ one) uses a key b. According to Definition 10 of autonomous keys, $Var_{\mathcal{I}}(b) = \emptyset$. Thus, V_1 remains stable and then V remains stable. The reasoning is the same for Rule 18. Rule 19 is applied when $\nexists Z_j \leqslant_m t$ and then V_2 does not change for this rule. V_1 remains stable since the rule preserve the same t. Rule 20 eliminates the block. Thus, V remains stable by the group G_3
Rule 21 eliminates the constraint. Rule 22 eliminates the whole block. Then, V remains stable or decreases. Rule 23 decomposes an equality constraint without generating new indexes. Then, V_3 remains stable. It does not generate other new constraints. Then, V is stable. Rule 24 generates new equality constraint $(u = w\delta_{l,k})$ where $Var_{\mathcal{I}}(u) = w\delta_{l,k} \subseteq \{k\}$ and $k \in Q$. Thus, V_3 remains stable and then V is stable. Therefore, V remains stable by the group G_4 . Rule 25 replace an index by another one which may lead to the generation of a new index for V. In this case, we must have in the block a constraint $(c_i = c_j)$ where $i, j \in R$, $i \in V$, $j \notin V$ and $ant(i, j) = j$. However, according to Invariant 10 for the constraint $(c_i = c_j)$, $ant(i, j) \in V$ which contradicts the fact that $j \notin V$. Thus, V is stable by Rule 25.
Rule 26 generates a new equality constraint $(u = v)$. If $\exists c_j$ s.t $j \in R$ and $c_j \leqslant_m u$ or $c_j \leqslant_m v$, then j belongs either to the constraint $(X_i = u)$ or $(X_i = v)$ and then V_3 does not change. The same reasoning is valid for Rule 27. Rule 29 can add an index to V_1. Nevertheless, this index comes from a sub-master constraint which is already computed in V_4. Then, V remains stable. The same reasoning is valid for Rule 31. Rule 30 does not generate new constraints. Rule 32 eliminates the whole block. Thus, V remains stable or decreases by the group G_5
Rule 33 is authorized only if $\nexists Z_j$ s.t $Z_j \leqslant_m t$. Then, V_2 does not change. Moreover, this rule generates a new Sub constraint from a Sub_d constraint preserving the same t. Then, V_1 is stable. Rule 35 is only authorized when $(X_m = v)$ and $\nexists Z_m$ s.t $Z_m \leqslant_m v$. Then, suppose this rule generates an equality constraint $(u_o\delta_o = v)$ that may add an index to V_3, i.e. $\exists Z_m, c_j$ s.t $j \in R$ and $Z_m \leqslant_m u_o\delta_o$. Then, j is already computed in V_3 thanks to the constraint $(X_m = v)$. Thus, V_3 remains stable. In the same way, the existential index that may be generated by the addition of the constraint $(v \in Forge_c(E'_r, \mathcal{K}_r))$ is already computed in V_3 thanks to the constraint $(X_m = v)$. Thus, $V_1 \cup V_3$ remains stable. Therefore, V is stable or decreases by the group G_6. □

We conclude that the set of indexes in V is bounded. □

Proposition 30. $Var^R_{\mathcal{I}}(\{u \mid t \in Sub(u, E, \mathcal{E}, \mathcal{K}) \vee t \in Sub_d(u, E, \mathcal{E}, \mathcal{K}) \in B\})$ is bounded.

This proposition will be used for the proof of Theorem 3, Proposition 31 and Proposition 32.

Proof. Considering a constraint of the form $(t \in Sub(v, E, \mathcal{E}, \mathcal{K}))$ or $(t \in Sub_d(v, E, \mathcal{E}, \mathcal{K}))$, since $Var_{\mathcal{I}}(t)$ is now bounded thanks to Proposition 29, we follow the same proof as the one of Proposition 27 in order to prove that
$Var^{\mathcal{X},R}_{\mathcal{I}}(\{v \text{ s.t } (t \in Sub(v, E, \mathcal{E}, \mathcal{K})) \text{ or } (t \in Sub_d(v, E, \mathcal{E}, \mathcal{K})) \in B \text{ and } \exists X_i \leqslant_m t\}) \cup Var^R_{\mathcal{I}}(\mathcal{E})$ is bounded. Then, we follow the same proof as the one of Proposition 28 in order to prove that
$Var^{\mathcal{C},R}_{\mathcal{I}}(\{v \mid (t \in Sub(v, E, \mathcal{E}, \mathcal{K})) \vee (t \in Sub_d(v, E, \mathcal{E}, \mathcal{K})) \in B)\}$ is bounded. We conclude that
$Var^R_{\mathcal{I}}(\{u \mid t \in Sub(u, E, \mathcal{E}, \mathcal{K}) \vee t \in Sub_d(u, E, \mathcal{E}, \mathcal{K}) \in B\})$ is bounded. □

Proposition 31. $Var^{\mathcal{X},R}_{\mathcal{I}}(\{u \mid (v = u) \vee (u = v) \in B\})$ is bounded.

This Proposition will be used for the proof of Theorem 3 and Proposition 32.

Proof. We prove Proposition 32 by proving the following invariant:
$Var^{\mathcal{X},R}_{\mathcal{I}}(\{u \mid (v = u) \vee (u = v) \in B\})$ is stable or increases by a set of indexes which is already bounded.
The first group does not generate new equality constraint with new indexes. Rules of the second group do not manage equality constraints. Then, the invariant is satisfied by G_1 and G_2.
The only rule of the third group that may generate an equality constraint is Rule 15. However, the possible existential indexes that may be in this equality constraint already belong to the Sub equality. Then, the set of these indexes is bounded according to Proposition 29 and Proposition 30 and thus, the invariant still holds. Rule 21 eliminates the constraint. Rule 22 eliminates the whole block. If Rule 24 generates a new index, then it would be an universal one. Thus, for these rules, the invariant is still satisfied. Rule 23 decomposes an equality constraint between two terms into another equality constraint between two subterms while preserving the same index variables. Then, the set of existential indexes remains stable and the invariant still holds. Rule 25 replaces an index by another using the constraint $(c_i = c_j)$. However, according to Invariant 10, $ant(i, j) \in V$. Then, even if Rule 25 lead to the generation of a new index in other equality constraints, then, this index belongs to a set which is already bounded since this index is in V and V is bounded according to

Proposition 29. Thus, the invariant still holds.
Only Rules 26 and 27 of the fifth group may generate new equality constraints. However, the possible indexes in these constraints already belongs to equality constraints in the left side of rules. Then, the set of indexes is stable and therefore the invariant is satisfied.
Only Rule 35 of the sixth group may generate new equality constraints: $(u_o\delta_o = v)$. If $\exists Z_j$ s.t $j \in R$ and $Z_j \leqslant_m v$, then according to Invariant 7, $j, m \in Q$ which is contradictory with $j \in R$. The same reason is valid for the case where $\exists Z_j$ s.t $j \in R$ and $Z_j \leqslant_m u_o\delta_o$ since u_o belongs to tha master constraint $(X_{i_o} = u_o)$. Then, the invariant still holds. □

Proposition 32. $Var_{\mathcal{I}}^{\mathcal{C},R}(\{u \,|\, (v = u) \vee (u = v) \in B\})$ is bounded.

Proof. We prove Proposition 32 by proving the following invariant:
$Var_{\mathcal{I}}^{\mathcal{C},R}(\{u \,|\, (v = u) \vee (u = v) \in B\})$ is stable or increases by a set of indexes that is already bounded.
We only focus on equality constraints $(v = u)$ that do not contain indexed variables. If $\exists X_i$ s.t $X_i \leqslant_m u$ or $X_i \leqslant_m v$ then $Var_{\mathcal{I}}^{\mathcal{C},R}(\{u \,|\, (v = u) \vee (u = v) \in B\})$ according to Proposition 29. Suppose then that $(v = u)$ does not contain indexed variables.

The first group does not generate new equality constraint with new indexes. Rules of the second group do not manage equality constraints. Then, the invariant is satisfied by G_1 and G_2.
The only rule of the third group that may generate an equality constraint is Rule 15. However, the possible existential indexes that may be in this equality constraint already belong to the Sub equality. Then, the set of these indexes is bounded according to Proposition 29 and Proposition 30 and thus, the invariant still holds. Rule 21 eliminates the constraint. Rule 22 eliminates the whole block. If Rule 24 generates a new index, then it would be an universal one. Thus, for these rules, the invariant is still satisfied. Rule 23 decomposes an equality constraint between two terms into another equality constraint between two subterms while preserving the same index variables. Then, the set of existential indexes remains stable and the invariant still holds. Rule 25 replaces an index by another using the constraint $(c_i = c_j)$. However, according to Invariant 10, $ant(i, j) \in V$. Then, even if Rule 25 lead to the generation of a new index in other equality constraints, then, this index belongs to a set which is already bounded since this index is in V and V is bounded according to Proposition 29. Thus, the invariant still holds.
Only Rules 26 and 27 of the fifth group may generate new equality constraints. However, the possible indexes in these constraints already belongs to equality constraints in the left side of rules. Then, the set of indexes is stable and therefore the invariant is satisfied.
Only Rule 35 of the sixth group may generate new equality constraints: $(u_o\delta_o = v)$. If $\exists c_j$ s.t $j \in R$ and $c_j \leqslant_m v$, then it was already counted since this index belongs to the constraint $(X_m = v)$ and according to Proposition 31, the set of such indexes is bounded. If $\exists c_j$ s.t $j \in R$ and $c_j \leqslant_m u_o\delta_o$, then we have a fresh index per indexed variable per master constraint. Thus, the number of possible existential indexes generated this way is bounded by (the number of indexed variable $*$ the number master constraint). However, the number of master constraints is bounded by the number of constraints in $\mathcal{E}$ which is fixed. Besides, the number of indexed variables is bounded by (the number of vectors $*$ the number of indexes in varibales in equality constraints). However, the number of vectors is defined by the specification of the protocol. Moreover, the number of indexes in variables in equality constraints is the sum of universal indexes and the existential ones. The set of universal indexes is bounded according to Proposition 26. The set of existential ones is bounded according to Proposition 31. We conclude that the invariant is satisfied by Rule 35. □

8.3 Termination For Protocols with Autonomous Keys

We first define a weight $\|\_\|$ for terms, elementary constraints, blocks and constraint systems. Then, we show that each rule decreases this weight.

In order to define the weight of a term, we need to introduce some definitions:

Definition 36 (Row of a term)**.** *A row of a term t is defined as follows:*

- $r(X) = max\{l \,|\, X \sqsubset_l Y, Y \in \mathcal{X} \cup \mathcal{X}_{\mathcal{I}}\}$ *for* $X \in \mathcal{X} \cup \mathcal{X}_{\mathcal{I}}$
- $r(t) = max\{r(Y) \,|\, Y < t, Y \in \mathcal{X} \cup \mathcal{X}_{\mathcal{I}}\}$

For a variable $X \in \mathcal{X} \cup \mathcal{X}_{\mathcal{I}}$, If $\nexists Y \in \mathcal{X} \cup \mathcal{X}_{\mathcal{I}}$, s.t $X \sqsubset Y$, then, $r(X) = 0$.

Definition 37 (Size of a term t)**.** *We define the size of a term t denoted by $|t|$ as follows:*

- $|t| = 1$ *for* $t \in \mathcal{X} \cup \mathcal{C}$
- $|f(u_1, .., u_m)| = 1 + |u_1| + .. + |u_m|$
- $|h(u)| = 2 + |u|$ *for* $h \in H$
- $|mpair(k, u)| = 2 + |u\delta|$ $\forall \delta$

We extend this definition to sets of terms by: $|E| = \Sigma_{t \in E}|t|$ *for* $E \subset T$.

Now, we can define the weight for terms, then, for elementary constraints and finally for constraint blocks and constraint system.

Definition 38 (Weight of a term t)**.** *Let p be the size of the protocol, i.e. the sum of sizes of messages. We define the weight of a term t denoted by $\|t\|$ as follows:*

- $\|X\| = p^{r(X)+1}$ *for* $X \in \mathcal{X} \cup \mathcal{X}_{\mathcal{I}}$
- $\|c\| = 1$ *for* $c \in \mathcal{C} \cup \mathcal{C}_{\mathcal{I}}$
- $\|f(u_1, .., u_m)\| = 1 + \|u_1\| + .. + \|u_m\|$
- $\|h(u)\| = 2 + \|u\|$ *for* $h \in H$
- $\|mpair(k, u)\| = 2 + \|u\delta\|$ $\forall \delta$

We extend this definition to sets of terms in the following way: $\|E\| = 1 + \Sigma_{t \in E}\|t\|$ *for* $E \subset T$.

Definition 39 (Weight of an elementary constraint)**.** *Let st be the number of the protocol subterms. We define the weight of an elementary constraint ctr denoted by $\|ctr\|$ as follows:*

- $\|t \in Forge(E, \mathcal{K})\| = \langle st - \#\mathcal{K},\ \|t\| + \|E\| + |E| + 1\rangle$
- $\|t \in Forge_c(E, \mathcal{K})\| = \langle st - \#\mathcal{K},\ \|t\| + \|E\| + |E|\rangle$
- $\|t \in Sub(w, E, \mathcal{E}, \mathcal{K})\| = \langle st - \#\mathcal{K},\ \|t\| + \|w\| + |E| + 1\rangle$
- $\|t \in Sub_d(w, E, \mathcal{E}, \mathcal{K})\| = \langle st - \#\mathcal{K},\ \|t\| + \|w\| + |E|\rangle$
- $\|t = u\| = \langle 0,\ \|t\| + \|u\|\rangle$

Before defining the weight of a constraint block, we first prove this proposition:

Proposition 33. *The number of application of Rule 26 is bounded.*

Proof. Rule 26 is applied once for each pair of constraints of the form $X_i = u$ and $X_i = v$ (thanks to $Hyp26$). However, according to Theorem 3, the set of indexes generated by our inference system is bounded. Then, the number of pairs of constraints for the same variable X_i is bounded. Thus, the number of application of Rule 26 is bounded. □

Definition 40 (Weight of a constraint block)**.** *We recall that, according to Theorem 3, the set of indexes generated by our inference system is bounded. Then, let Na be the maximum number of application of Rule 26 (See Proposition 33, this number is bounded). Let NcE be the maximum number of* Equality *constraints of the form $X = u$ where $X \in X$, $NcEF$ be the maximum number of constraints of the form $X = u$ or $X \in Forge_c(E, \mathcal{K})$ where $X \in \mathcal{X} \cup \mathcal{X}_{\mathcal{I}}$ and NcN be the maximum number of* Negative *constraints of type $Forge$ or equality.*

Consider a constraint block: $B = ctr_1 \wedge ..ctr_l$. We denote by Fc_B the number of final constraints in B, Nc_B the number of of negative constraints in B, Sc_B the number of submaster constraints in B, MEc_B the number of master equality constraints in B, MFc_B the number of master constraints in B of the type $Forge$, Ec_B the number of equality constraints of the form $X = u$ ($X \in \mathcal{X} \cup \mathcal{X}_{\mathcal{I}}$) in B and NR_B the number of application of Rule 26. We denote also the lexicographic order by $\langle\rangle$ and the multiset by $[]$. We define the weight of B as follows:

$$\|B\| = \langle Na - NR_B,\ NcEF - Fc_B,\ NcN - Nc_B,\ NcE - Sc_B,\ NcEF - MEc_B,\\ NcEF - MFc_B,\ [\|ctr_i\|]_{ctr_i \in B},\ NcE - Ec_B\rangle$$

Definition 41 (Weight of a constraint System)**.** *Consider a constraint system:* $S = B_1 \wedge ..B_l$*. We define the weight of* S *as follows:*

$$\|S\| = [\|B_i\|]_{B_i \in S}$$

Proposition 34. *Algorithm 1 terminates for protocols with autonomous keys.*

Proof. We prove Proposition 34 by showing that each rule of our inference system decreases the weight of the whole constraint system S. That is, considering a rule r, we show that $\|\text{post}(r)\| < \|\text{pre}(r)\|$. We denote always the block treated by these rules B. First, Rule 26 decreases the size of the block for which it is applied since it decreases $Na - NR_B$. Then, Rules 3, 14, 20, 22, 32 eliminate one block of the constraint system since they lead to $\bot$. Thus, they decrease $\|S\|$. Moreover, Rules 21, 25 and 30 eliminate a constraint from the block treated B since they lead to $\top$. Then, $\|B\|$ decreases.

Rule 1 changes an equality constraint $ctr \in B$ into another one of the form $(X = u)$ where $X \in X \cup \mathcal{X}_\mathcal{I}$. The only parameter that changes for the weight of B is Ec_B which increases and then $NcE - Ec_B$ decreases. Then, $\|B\|$ decreases and so does $\|S\|$.

For Rule 2, we have $\|X\| > \|u\|$ since $X = u \in B$. Indeed, $\|u\| \leq |u| * p^{max\{\mathrm{r}(Y)|Y<u, Y \in \mathcal{X} \cup \mathcal{X}_\mathcal{I}\}} \leq p^{\mathrm{r}(X)+1}$. Then, $\|Y = X\| = \|Y\| + \|X\| > \|Y\| + \|u\| = \|Y = u\|$. Thus, $\|Y = X\| > \|Y = u\|$ and therefore $\|B\|$ decreases and so does $\|S\|$.

Rule 4 adds a new master constraint and changes only dull constraints. Then, the only parameter that changes for the weight of B is either $NcEF - MEc_B$ or $NcEF - MFc_B$. Then, $\|B\|$ decreases and so does $\|S\|$.

Rule 5 transforms a master constraint of the form $Forge$ to a master equality constraint. Then, it increases $NcEF - MFc_B$ but decreases $NcEF - MEc_B$. Then, $\|B\|$ decreases and so does $\|S\|$.

For Rule 6, we have $\|u\lambda\| < \|u\|$ where $\lambda = [Y_j \leftarrow Z]$. Indeed, since $Y_j = Z \in B$, we have $\mathrm{r}(Z) < \mathrm{r}(Y_j)$ which leads to $\|Z\| < \|Y_j\|$. Then, $\|X = u\lambda\| < \|X = u\|$ and therefore $\|B\|$ decreases.

For Rule7, we reason similarly to Rule 6 to prove that $\|X = u\lambda\| < \|X = u\|$. Moreover, $\|Y_j = Z\| < \|X = u\|$. Indeed, $Y_j < u$. then, $\|Y_j\| < \|u\|$. Besides, Z is a fresh variable and since $X = u \in B$ then $\mathrm{r}(Z) < \mathrm{r}(X)$. Thus, $\|Z\| < \|X\|$ and therefore $\|Y_j = Z\| < \|X = u\|$. We conclude that $\|B\|$ decreases since the other parameters do not change.

Rule 8 adds a submaster constraint to the block B. Then, $NcE - Sc_B$ decreases and therefore $\|B\|$ decreases.

For Rule 9, $\|\text{pre}(R\ 9)\| > \|\text{post}(R\ 9)\|$. Indeed, $\|\text{pre}(R\ 9)\| = \langle k, \|t\| + \|E\| + |E| + 1\rangle$ and $\|\text{post}(R\ 9)\| = [\langle k, \|t\| + \|E\| + |E|\rangle, \langle k, \|t\| + \|w\| + |E| + 1\rangle]$, where $k = st - \#\mathcal{K}$ and $w \in E$ and therefore $\|w\| < \|E\|$. Moreover, Rule 9 do not change the other parameters of the weight of B. Then, $\|B\|$ decreases.

For Rule 10, $\|\text{pre}(R\ 10)\| = \langle k, 1 + \|t_1\| + .. + \|t_m\| + \|E\| + |E|\rangle$ where $k = st - \#\mathcal{K}$. Moreover, $\|\text{post}(R\ 10)\| = [\langle k, \|t_i\| + \|E\| + |E| + 1\rangle]_{t_i < \langle t_1..t_m\rangle}$. Then, $\|\text{pre}(R\ 10)\| > \|\text{post}(R\ 10)\|$. Besides, Rule 10 do not change the other parameters of the weight of B. Then, $\|B\|$ decreases.

We show in a similar way as Rule 10 that for Rule 11, $\|B\|$ decreases.

For Rule 12, $\|\text{pre}(R\ 12)\| > \|\text{post}(R\ 12)\|$. Indeed, $\|\text{pre}(R\ 12)\| = \langle k, 2 + \|t\| + \|E\| + |E|\rangle$ and $\|\text{post}(R\ 12)\| = \langle k, 1 + \|t\| + \|E\| + |E|\rangle$, where $k = st - \#\mathcal{K}$. Moreover, Rule 12 do not change the other parameters of the weight of B. Then, $\|B\|$ decreases.

We show in a similar way as Rule 13 that for Rule 12, $\|B\|$ decreases.

For Rule 15, we have $\|\text{pre}(R\ 15)\| = \langle k, 1 + \|t\| + \|u\| + |E|\rangle$ and two cases for post$(R\ 15)$: either $\|\text{post}(R\ 15)\| = \langle k, \|t\| + \|u\|\rangle$ or $\|\text{post}(R\ 15)\| = [\langle k, \|t\| + \|u\|\rangle, \langle k, \|t\| + \|u\| + |E|\rangle]$. In both cases, $\|\text{pre}(R\ 15)\| > \|\text{post}(R\ 15)\|$. Besides, since this rule do not change the other parameters of the weight of B. Then, $\|B\|$ decreases.

We show in a similar way as Rule 10 that for Rule 16, $\|B\|$ decreases.

For Rule 17, first, $\|\text{pre}(R\ 17)\| = \langle k, 1 + \|t\| + \|u\| + \|b\| + |E|\rangle$ where $k = st - \#\mathcal{K}$. Second, $\|\text{post}(R\ 17)\| = [\langle k, 1 + \|t\| + \|u\| + |E|\rangle, \langle k', 1 + \|b\| + \|E\| + |E|\rangle]$ where $k' = st - \#\mathcal{K} \cup \{\{u\}_p^p\}$. Since $k' < k$, $\|\text{pre}(R\ 17)\| > \|\text{post}(R\ 17)\|$.

We show in a similar way as Rule 17 that for Rule 18, $\|S\|$ decreases.

For Rule 19, first, $\|\text{pre}(R\ 19)\| = \langle k, 2 + \|t\| + \|u\delta\| + |E|\rangle$ where $k = st - \#\mathcal{K}$. Second, $\|\text{post}(R\ 19)\| = \langle k, 1 + \|t\| + \|u\delta\| + |E|\rangle$. Then, $\|\text{pre}(R\ 19)\| > \|\text{post}(R\ 19)\|$. Since this rule do not change the other parameters of the weight of B. Then, $\|B\|$ decreases.

For Rule 23, first, $\|\text{pre}(R\ 23)\| = \langle k, 2 + \|u_1\| + .. + \|u_m\| + \|w_1\| + .. + \|w_m\|\rangle$. Second, $\|\text{post}(R\ 23)\| = [\langle k, \|u_i\| + \|w_i\|\rangle]_{i=1..m}$ where $k = st - \#\mathcal{K}$. Then $\|\text{pre}(R\ 23)\| > \|\text{post}(R\ 23)\|$ and since the other parameters of the weight of B do not change, $\|B\|$ decreases.

For Rule 24, $\|\text{pre}(R\ 24)\| > \|\text{post}(R\ 24)\|$. Indeed, $\|\text{pre}(R\ 24)\| = \langle k, 4 + \|u\delta\| + \|w\delta_{l,k}\delta\|\rangle$. Moreover, $\|\text{post}(R\ 24)\| = \langle k, \|u\delta\| + \|w\delta_{l,k}\delta\|\rangle$. Then, since the other parameters of the weight of B do not change, $\|B\|$ decreases.

We show in a similar way as Rule 2 that for Rule 27 $\|B\|$ decreases since $\|X\| > \|u\|$ and therefore $\|X = v\| > \|u = v\|$.

We show in a similar way as Rule 27 that for Rule 28 $\|B\|$ decreases since $X_i = u \in B$ and then $\|X_i\| > \|u\|$ and therefore $\|X_i \in Forge_c(E', \mathcal{K})\| > \|u \in Forge_c(E', \mathcal{K})\|$. The same proof is valid for Rule 29.

For Rule 31, since $X = w \in B$, $\|X\| > \|w\|$. Then, $\|t \in Sub_d(X, E', \mathcal{E}, \mathcal{K})\| > \|t \in Sub_d(w, E', \mathcal{E}, \mathcal{K})\|$. Besides, this rule do not change the other parameters of the weight of B. Then, $\|B\|$ decreases.

Rule 33 adds a new final constraint to the block treated B which decreases $NcEF - Fc_B$. Then, $\|B\|$ decreases. Rule 34 either adds a negative constraint to the bloc treated B or adds a final constraint to B. In both cases, $\|B\|$ decreases. In a similar way, Rule 35 decreases $\|B\|$. □

9 Checking Satisfiability

Let $F = \forall Q \exists R\, B_1 \vee .. \vee B_p$ be a normalized constraint system as created by our decision algorithm. We choose two index variables q and r, fixed for all the following, that we will use as replacement for index in Q or R respectively. We rewrite here the definitions given earlier and give precise the application of rules $E1$ and $E2$. First, let us define patterns of a term and a constraint system :

Definition 42. *We say that u is a pattern of v for index variable i, denoted $u \ll_i v$, iff $u \leqslant_m v$ and $Var_{\mathcal{I}}(v) \subset \{i\}$ or there exists $mpair(k, t) \leqslant_m v$ such that $u \ll_i t\delta_{k,i}$.*

Definition 43. *Given a constraint system $F = \forall Q \exists R\, B_1 \vee .. \vee B_p$, let $\bar{\delta_q}$ be $\delta_{Q,R\to q}$ and $\bar{\delta_r}$ be $\delta_{Q,R\to r}$. We denote by $Patt(F)$ the set of all patterns in F defined by :*

$$Patt(F) = \left\{u \,|\, \exists v \in \mathcal{T} \text{ in } F \text{ s.t. } u \notin \mathcal{X}_{\mathcal{I}},\ Var^{\mathcal{X}}_{\mathcal{I}}(u) \subseteq \{q\} \text{ and } u \ll_q v\bar{\delta_q} \text{ or } u \ll_r v\bar{\delta_r}\right\}$$

Second, we choose an order on $Patt(F)$, denoted $\lhd$ and fixed for all the following. For any $u \in \mathcal{T}$, we denote by $u\bar{\delta}$ the higher term in $Patt(F)$ between $u\bar{\delta_q}$ and $u\bar{\delta_r}$ w.r.t $\lhd$. Also, we define two extra deduction rules $E1$ and $E2$ ensuring that our system always uses the lower patterns w.r.t $\lhd$ with higher priority:

$$(X_m = u)^{\cdot} \rightarrow \bigvee_{u' \lhd u\bar{\delta}} \exists k. \left((X_m = u'\delta^k_{q,m})^{\cdot} \wedge \bigwedge_{v \lhd u'} \left(\forall k'. X_m \neq v\delta^{k'}_{q,m}\right) \wedge u'\delta^k_{q,m} = u\right) \quad (E1)$$

$$(X_m \in Forge_c(E, \mathcal{K}))^{\cdot} \rightarrow \bigvee_{u' \in Patt(F)} \exists k. \left((X_m = u'\delta^k_{q,m})^{\cdot} \wedge \bigwedge_{v \lhd u'} \left(\forall k'. X_m \neq v\delta^{k'}_{q,m}\right) \wedge u'\delta^k_{q,m} \in Forge_c(E, \mathcal{K})\right) \quad (E2)$$
$$\vee (X_m \in Forge_c(E, \mathcal{K}))^{\cdot} \wedge \bigwedge_{v \in Patt(F)} \left(\forall k'. X_m \neq v\delta^{k'}_{q,m}\right)$$

To prevent cycles, the rule $E1$ cannot be used if the block already contains $\bigwedge_{v \lhd u\bar{\delta}} \left(\forall k'. X_m \neq v\delta^{k'}_{q,m}\right)$, and the rule $E2$ cannot be used if the block already contains $\bigwedge_{v \in Patt(F)} \left(\forall k'. X_m \neq v\delta^{k'}_{q,m}\right)$. Moreover, $E1$ and $E2$ implicitly update the environment $\mathcal{E}$ of each block in the constraint system by recomputing them the same way as in Algorithm 1, except that only variables present in some $Forge_c(E, \mathcal{K})$ in F can get values in $\mathcal{E}$. This way, we can normalize again any constraint system computed by $E1$ or $E2$ using the phase 2 of the normalization from definition 25. That is, we denote by $\downarrow^{Ext}$ the application of as many rules $E1$ and $E2$ as possible (minimum one), followed by the phase 2 of the normalization. However, the normalisation may create new constraints for which we need $E1$ and $E2$ again, and thus, we need to iterate $\downarrow^{Ext}$. Hopefully, we remark that :

Lemma 12. *$Patt(F) = Patt(F \downarrow^{Ext})$, and $[\![F]\!]^e_{\emptyset} = \left[\!\left[F \downarrow^{Ext} \right]\!\right]^e_{\emptyset}$.*

Proof. This follows directly from the definition of patterns, which includes subterms, and from the rules which always extract subterms, i.e. nothing else than subterms of patterns can appear during the normalization. More precisely, $Patt(F)$ is stable by application of any rule since :

- Implicit reformation rules have no effect on patterns;
- Prioritary rules do not create new patterns;
- Forge, Sub or equality reduction rules : create new constraints with only patterns inherited from previous ones;
- Interleaving rules : do not create new patterns;
- $E1$ and $E2$ rules : do not create new patterns.

which shows the stability of $Patt(..)$. Moreover, the rules $E1$ and $E2$ naturally preserves the semantics, since it creates disjunctions based on an exhaustive enumeration of all possible values of X_m. Details are similar to the correction and completeness of the block interleaving rules. This proves the lemma. □

Proposition 35. *If $\downarrow^{Ext}$ iterated on CBS_{k+1} terminates, then there exists e_{max} computable such that :*

$$\forall e \geq e_{max},\ \ [\![CBS_{k+1}]\!]^e_\emptyset = [\![CBS_{k+1}]\!]^{e_{max}}_\emptyset$$

Proof. As above, let $F = \forall Q \exists R\, B_1 \vee .. \vee B_p$ be a normalized constraint system as created by our decision algorithm, i.e. $F = CBS_{k+1}$, and let q and r be the two index variables fixed at the beginig of this section. Using lemma 12 and hypothesis on $\downarrow^{Ext}$, we can iterate $\downarrow^{Ext}$ until a constraint system on which no $E1$ or $E2$ rule can be applied, that is, a system where :

- Any constraint $(X_m = u)^\cdot$ comes with $\bigwedge_{v \lhd u\bar{\delta}} \left(\forall k'. X_m \neq v\delta^{k'}_{q,m}\right)$ in the same block;
- Any constraint $(X_m \in Forge_c(E, \mathcal{K}))^\cdot$ comes with $\bigwedge_{v \in Patt(F)} \left(\forall k'. X_m \neq v\delta^{k'}_{q,m}\right)$ in the same block.

Let F' be this constraint system obtained from F by iteration of $\downarrow^{Ext}$.

Third, we transform F' into a system where we choosed values for any variable under a forge constraint. Let c be a ground term in E_0, i.e a fixed message that the intruder knows from the start. Let $d = \sum_{(X=u)^\cdot \in F'} Depth(u)$ be the sum of all depth of equalities in F'. Note that d is bounded by the size of F'. For any $\overrightarrow{X} \in \overrightarrow{\mathcal{X}}$ we choose a ground term $b_{\overrightarrow{X}} = \langle c, .., c \rangle$ such that $\forall \overrightarrow{X}, \overrightarrow{Y}$, $b_{\overrightarrow{X}} \neq b_{\overrightarrow{Y}}$. Now, for any $\overrightarrow{X} \in \overrightarrow{\mathcal{X}}$ let $t_{\overrightarrow{X}} = \left\{.. \left\{b_{\overrightarrow{X}}\right\}_c ..\right\}_c$ be the term of depth $(d+1)$. Note that by construction, $\forall \overrightarrow{X}, \overrightarrow{Y}$, $t_{\overrightarrow{X}} \neq t_{\overrightarrow{Y}}$ and $\forall \overrightarrow{X}$, $t_{\overrightarrow{X}} \in DY_c(E_0, \emptyset)$. Now, let F'' be the closure of F' by :

$$(X_i \in Forge_c(E))^\cdot \rightarrow \left(X_i = t_{\overrightarrow{X}}\right)^\cdot \qquad (E3)$$

Lemma 13. *If $\exists \sigma \in [\![F']\!]^e_\emptyset$ then $\exists \sigma' \in [\![F'']\!]^e_\emptyset$.*

Proof. Here, we do not guaranty that any solution of F' is preserved, but only that at last one survives. That is, let $\sigma \in [\![F']\!]^e_\emptyset$ with $F' = \forall Q \exists R\, B_1 \vee .. \vee B_p$. We define σ' such that :

- $\forall X \in \mathcal{X}$, $\sigma'(X) = \sigma(X)$;
- $\forall \overrightarrow{X} \in \overrightarrow{\mathcal{X}}$, $\forall s \in \{1..e\}$, if $\exists u \in Patt(F')$, $\exists \tau$ with $\tau(q) = s$ such that $\sigma(X_s) = \sigma(u\tau)$ and u minimal w.r.t $\lhd$, then $\sigma'(X_s) = \sigma'(u\tau)$;
  Note that u is single by minimality w.r.t $\lhd$ and because $Var^{\mathcal{X}}_{\mathcal{I}}(u) \subseteq \{q\}$.
- $\forall \overrightarrow{X} \in \overrightarrow{\mathcal{X}}$, $\forall s \in \{1..e\}$, if $\forall u \in Patt(F')$, $\forall \tau$ with $\tau(q) = s$, we have $\sigma(X_s) \neq \sigma(u\tau)$, then $\sigma'(X_s) = t_{\overrightarrow{X}}$.

We show that $\sigma' \in [\![F'']\!]^e_\emptyset$: Thanks to $\sigma \in [\![F']\!]^e_\emptyset$, we know that $\forall \tau_Q$, $\exists \tau \supseteq \tau_Q$, $\exists i \in \{1..p\}$, $\forall ctr \in B_i$ $\sigma \in [\![ctr]\!]^e_\tau$. We examine all possible constraints ctr :

- If $ctr = (X = u)^{sm}$ or $ctr = (X \in Forge_c(E, \mathcal{K}))$, then $\sigma' \in [\![ctr]\!]^e_\tau$ since $\sigma'(X) = \sigma(X)$;

- If $ctr = (X_m = u)^{\bullet}$, then thanks to the definition of F', we have also $\sigma \in \left|\left[\bigwedge_{v \lhd u\bar{\delta}} \left(\forall k'.X_m \neq v\delta_{q,m}^{k'}\right) \right]\right|_{\tau}^{e}$. Therefore, $u' = u\bar{\delta_q}$ or $u' = u\bar{\delta_r}$, with $Var_{\mathcal{I}}^{\mathcal{X}}(u') \subseteq \{q\}$, is a pattern $u' \in Patt(F)$. Moreover, it is the minimal pattern w.r.t $\lhd$ (since $Var_{\mathcal{I}}^{\mathcal{X}}(u) \subseteq \{m\}$) such that $\sigma(X_m\tau) = \sigma(u'\tau')$ with $\tau'(q) = \tau(m)$, $\tau'(r) = \tau(k)$ and $k \in Var_{\mathcal{I}}(u) \backslash Var_{\mathcal{I}}^{\mathcal{X}}(u)$ if non-empty. Thus, u' matches the second point in the definition of σ', i.e. $\sigma'(X_m\tau) = \sigma'(u'\tau') = \sigma'(u\tau)$ and $\sigma' \in [\![ctr]\!]_{\tau}^{e}$.
- If $ctr = (X_m \in Forge_c(E, \mathcal{K}))^{\bullet}$, then again thanks to the definition of F', we have $\sigma \in \left|\left[\bigwedge_{v \in Patt(F)} \left(\forall k'.X_m \neq v\delta_{q,m}^{k'}\right) \right]\right|_{\tau}^{e}$. Therefore, $\sigma'(X_m\tau) = t_{\overrightarrow{X}}$ and thus, $\sigma' \in \left|\left[X_m = t_{\overrightarrow{X}} \right]\right|_{\tau}^{e}$.
- If $ctr = (\forall k\, X_m \neq u)$, we prove by recurrence a more general property of σ and σ'. We do this in two steps :

  Claim 2. $\forall u \in Patt(F)$, $\forall \tau$, $\forall \overrightarrow{Z}$, $t_{\overrightarrow{Z}} \neq \sigma'(u\tau)$

  Proof. We know that the value of some X_s through σ' is a pattern using only variables indexed by s. Therefore, we have only two cases : either there exists some $t_{\overrightarrow{Y}}$ subterm of $\sigma'(u\tau)$, and thus $Depth(\sigma'(u\tau)) > d+1$ since $u \notin \mathcal{X}_{\mathcal{I}}$, or there is no such $t_{\overrightarrow{X}}$ subterm of $\sigma'(u\tau)$, and thus $Depth(\sigma'(u\tau)) \leq d$ since the same patterns cannot appear twice in the same branch of $\sigma'(u\tau)$ without creating a cycle. In both cases $Depth(\sigma'(u\tau)) \neq d+1$ which proves the claim. □

  Claim 3. $\forall w, v \in Patt(F) \cup \mathcal{X}_{\mathcal{I}}$ *with* $u = w\delta_{r,r'}$, $\forall \tau$ *if* $\sigma(u\tau) \neq \sigma(v\tau)$ *then* $\sigma'(u\tau) \neq \sigma'(v\tau)$.

  Proof. Assume that the property is true for $Depth(u) + Depth(v) < n$, n integer. Let u and v be two terms as above with $Depth(u) + Depth(v) = n$. We have cases depending of the structure of u and v :

  - If $u = f(\{u_i\})$ and $v = g(\{v_j\})$ then either $f \neq g$ and thus $\sigma'(u\tau) \neq \sigma'(v\tau)$, or $f = g$ and $\exists i$ $\sigma(u_i\tau) \neq \sigma(v_i\tau)$ with $Depth(u_i) + Depth(v_i) < n$, i.e. $\sigma'(u_i\tau) \neq \sigma'(v_i\tau)$ and thus $\sigma'(u\tau) \neq \sigma'(v\tau)$. This case includes constants, i.e. f of g with no parameter.
  - If $u = Y_q$ and $v = Z_q$ with $\sigma'(Y_q\tau) = \sigma'(u'\tau)$ and $\sigma'(Z_q\tau) = \sigma'(v'\tau)$, it means that $\sigma(Y_q\tau) = \sigma(u'\tau)$ and $\sigma(Z_q\tau') = \sigma(v'\tau)$, with $\sigma(u'\tau) \neq \sigma(v'\tau)$. Thus, similarly as in the case above, we have $\sigma'(u'\tau) \neq \sigma'(v'\tau)$ and thus $\sigma'(u\tau) \neq \sigma'(v\tau)$.
  - If $u = Y_q$ and $v = Z_q$ with $\sigma'(Y_q\tau) = t_{\overrightarrow{Y}}$ and $\sigma'(Z_q\tau) = t_{\overrightarrow{Z}}$, then $t_{\overrightarrow{Y}} \neq t_{\overrightarrow{Z}}$ and thus $\sigma'(Y_q\tau) \neq \sigma'(Z_q\tau)$. Note that $u = v$ is impossible.
  - If $u = Y_q$ and $v \notin \mathcal{X}_{\mathcal{I}}$ with $\sigma'(Y_q\tau) = t_{\overrightarrow{Y}}$, then thanks to claim 2 above we have $\sigma'(Y_q\tau) = t_{\overrightarrow{Y}} \neq \sigma'(v\tau)$. The same holds with u and v reversed.
  - If $u = Y_q$ and $v = Z_q$ with $\sigma'(Y_q\tau) = t_{\overrightarrow{Y}}$ and $\sigma'(Z_q\tau) = \sigma'(v'\tau)$, then thanks to claim 2 again we have $\sigma'(Y_q\tau) = t_{\overrightarrow{Y}} \neq \sigma'(v'\tau)$.

  This proves the claim. □

  Consequently, since $ctr = (\forall k\, X_m \neq u)$ we have $\sigma(X_m\tau') \neq \sigma(u\tau')$ with any $\tau' \supseteq \tau$ such that $Dom(\tau') = Dom(\tau) \cup \{k\}$, and thus thanks to claim 2, $\sigma'(X_m\tau') \neq \sigma'(u\tau')$ i.e. $\sigma' \in [\![ctr]\!]_{\tau}^{e}$.
- If $ctr = (X_m \notin Forge_c(E))$, assuming $\overline{\mathcal{X}} = \left\{u \in \mathcal{X} \cup \mathcal{X}_{\mathcal{I}} \,|\, Var_{\mathcal{I}}^{\mathcal{X}}(u) \subseteq \{q\}\right\}$, we prove two more generic properties by iteration:

  Claim 4. $\forall u \in Patt(F)$, $\forall \tau$, *let d be (one of) the minimal derivation proving* $\sigma'(u\tau) \in DY(E'\sigma', \overline{\mathcal{K}}^{e}\tau\sigma')$ *s.t. no term in* $\left\{t_{\overrightarrow{X}}\right\}_{\overrightarrow{X} \in \overrightarrow{\mathcal{X}}}$ *is decomposed, with* $E' = E \cup \left\{t_{\overrightarrow{X}}\right\}$. *Then* $\forall t$ *knowledge in d,* $\exists v \in Patt(F) \cup \overline{\mathcal{X}}$ $\exists \tau'$ *s.t.* $t = \sigma'(v\tau')$. *Moreover, replacing each t by* $\sigma(v\tau')$ *and* E' *by* $E'' = E \cup \{\sigma(X_s) \,|\, \forall w \in Patt(F), \forall \tau', \sigma(X_s) \neq \sigma(w\tau')\}$ *in d maintains the validity of each rule's application.*

  Proof. We uses the structure of any minimal derivation, i.e. that any decomposed term is a subterm of the initial knowledge, and that any composed term is a subterm of either the goal or a decomposed term (otherwise there would be useles rules in the derivation). Knowing this, we remark that :

- Let $L = L_d(\sigma'(w\tau')) \in d$ such that $w \in Patt(F) \cup \overline{\mathcal{X}}$. Let t be the term generated by L. Then either $w = f(\{w_i\})$, and thus $\exists i$ such that $t = \sigma'(w_i\tau')$, i.e. $\exists w' \in Patt(F) \cup \overline{\mathcal{X}}\ \exists \tau''$ such that $t = \sigma'(w'\tau'')$, since subterm of patterns are patterns too. Note that $\tau'' = \tau'$ unless f is an mpair; Or $w \in \mathcal{X} \cup \mathcal{X}_{\mathcal{I}}$, and thus $\forall \overrightarrow{X}$, $\sigma'(w\tau') \neq t_{\overrightarrow{X}}$ since such terms cannot be decomposed in d. Therefore, $\exists v \in Patt(F)\ \exists \tau''$ such that $\sigma'(w\tau') = \sigma'(v\tau'')$, and since $v \notin \mathcal{X} \cup \mathcal{X}_{\mathcal{I}}$ we can repeat the same reasonning as with $w = f(\{w_i\})$.
- Let $L = L_c(\sigma'(w\tau')) \in d$ such that $w \in Patt(F) \cup \overline{\mathcal{X}}$. Let $\{t_i\}$ be the set of terms used by L to generate $\sigma'(w\tau')$. Then either $w = f(\{w_i\})$, and thus $\forall i\ \exists w'_i \in Patt(F) \cup \overline{\mathcal{X}}\ \exists \tau''_i$ such that $t_i = \sigma'(w'_i\tau''_i)$; Or $w \in \mathcal{X} \cup \mathcal{X}_{\mathcal{I}}$ with $\exists v \in Patt(F)\ \exists \tau''$ such that $\sigma'(w\tau') = \sigma'(v\tau'')$, and thus the same reasoning as for $w = f(\{w_i\})$ applies; Note that $w \in \mathcal{X} \cup \mathcal{X}_{\mathcal{I}}$ with $\exists \overrightarrow{X}$ such that $\sigma'(w\tau') = t_{\overrightarrow{X}}$ is impossible by minimality, since each $t_{\overrightarrow{X}}$ is already in E'.

Therefore, the first part of the claim follows by iteration of these two steps on the structure of d, starting from $E\sigma$ or $\sigma'(u\tau)$. Moreover, we remark that all the patterns we chosed for knowledges in d validates the subterm property for each rule, modulo renaming of index variables: that is, a decomposition rule generates a term with a pattern that is a subterm of the pattern of it's argument, module index renaming, and reversly for composition rules. Therefore, replacing σ' by σ cannot change the validity of each rule's application, provided that the values of variables with no pattern are still available in the initial knowledge (i.e. E'' replacing E'), which proves the claim. □

We uses this claim on $ctr = (X_m \notin Forge_c(E, \mathcal{K}))$, i.e. assume that $\sigma'(X_m\tau) \in DY_c(E\sigma', \overline{\mathcal{K}}^e\tau\sigma')$ despites the fact that $\sigma(X_m\tau) \notin DY_c(E\sigma, \overline{\mathcal{K}}^e\tau\sigma)$. Then there exists a derivation d defined as in Claim 4 with $E' = E \cup \{t_{\overrightarrow{X}}\}$, since $\forall \overrightarrow{X}\ t_{\overrightarrow{X}} \in DY_c(E\sigma', \overline{\mathcal{K}}^e\tau\sigma')$, and thus d holds the same properties. Therefore, there exists a (valid) derivation d' obtained from d as described above, starting from $E''\sigma$, with goal $\sigma(u\tau)$, and with the same structure as d i.e. finishing by a composition rule. Moreover, $E''\backslash E$ contains only terms that that can be forged with (non-redundant) derivations finishing by a composition rule, too. Therefore, it follows that $\sigma(X_m\tau) \in DY_c(E\sigma, \overline{\mathcal{K}}^e\tau\sigma)$, thus contradicting $\sigma(X_m\tau) \notin DY_c(E\sigma, \overline{\mathcal{K}}^e\tau\sigma)$. Consequently, the hypothesis was false, i.e. $\sigma'(X_m\tau) \notin DY_c(E\sigma', \overline{\mathcal{K}}^e\tau\sigma')$ and thus, $\sigma' \in [\![ctr]\!]^e_\tau$.

Since there are no other kind of constraint in a normalized constraint system, this proves the lemma, i.e. $\sigma' \in [\![F'']\!]^e_\emptyset$. □

Moreover, we remark that naturally, if $\sigma' \in [\![F'']\!]^e_\emptyset$ then $\sigma' \in [\![F']\!]^e_\emptyset$, since $X_i = t_{\overrightarrow{X}}$ is strictly more restrictive than $X_i \in Forge_c(E)$. Therefore, we have the following property :

$$[\![F']\!]^e_\emptyset \neq \emptyset \text{ iff } [\![F'']\!]^e_\emptyset \neq \emptyset$$

and thus, it is enough to check the satisfiability of F'' instead of F'. However, F'' has a very interesting property that will allow us to bound e with equivalent satisfiability. That is :

$$\forall \sigma \in [\![F'']\!]^e_\emptyset, \forall s \in \{1..e\}, \forall \overrightarrow{X} \in \overrightarrow{\mathcal{X}}, \quad \begin{array}{l}\text{either } \sigma(X_s) = t_{\overrightarrow{X}} \\ \text{or } \exists u \in Patt(F)\ \exists \tau \text{ with } \tau(q) = s \text{ s.t. } \sigma(X_s) = \sigma(u\tau)\end{array} \qquad (P1)$$

Note that this structure comes from rules $E1$, $E2$ and $E3$. From now on, let us fix some $\sigma \in [\![F'']\!]^e_\emptyset$ (assuming there exists one), some pattern $u_{X_s} \in Patt(F) \cup \{t_{\overrightarrow{X}}\}$ and τ_{X_s} such that $\tau_{X_s}(q) = s$ and $\sigma(X_s) = \sigma(u_{X_s}\tau_{X_s})$, according to $P1$. We remark that since $Patt(F)$ and $\overrightarrow{\mathcal{X}}$ are finite, i.e. bounded by a function in the size of the protocol specification, there exists necessarily only a bounded number of possible choices for $\{u_{X_s}\}_{\overrightarrow{X} \in \overrightarrow{\mathcal{X}}}$, independently of s. Let e_1 be this bound. Moreover, for each X_s there is also only a bounded number of set of knowledges E from F'' and keys $\mathcal{K}$ such that $\sigma(X_s) \in DY_c(E\sigma, \mathcal{K}\sigma)$. Let e_2 be this bound. We write $e_{max} = e_1 \times e_2 \times \#\overrightarrow{\mathcal{X}}$. Naturally, if $e > e_{max}$ then there exists $s \neq s'$ such that $\{u_{X_s}\}_{\overrightarrow{X} \in \overrightarrow{\mathcal{X}}} = \{u_{X_{s'}}\}_{\overrightarrow{X} \in \overrightarrow{\mathcal{X}}}$ and $\forall (\_ \in Forge_c(E, \mathcal{K}))$ in F'', $\forall \overrightarrow{X}$, $\sigma(X_s) \in DY_c(E\sigma, \mathcal{K}\sigma)$ iff $\sigma(X_{s'}) \in DY_c(E\sigma, \mathcal{K}\sigma)$. We will use this to prove that there is no need to search for attacks with $e > e_{max}$, i.e. :

Lemma 14. *If $e > e_{max}$ and $\sigma \in [\![F'']\!]^e_\emptyset$, then $\exists \sigma' \in [\![F'']\!]^{e-1}_\emptyset$.*

Proof. Let s, s' be as defined above, i.e. s.t. $\{u_{X_s}\}_{\overrightarrow{X}\in\overrightarrow{\mathcal{X}}} = \{u_{X_{s'}}\}_{\overrightarrow{X}\in\overrightarrow{\mathcal{X}}}$ and $\forall\,(\_ \in Forge_c(E,\mathcal{K}))$ in F'', $\forall\overrightarrow{X}$, $\sigma(X_s) \in DY_c(E\sigma,\mathcal{K}\sigma)$ iff $\sigma(X_{s'}) \in DY_c(E\sigma,\mathcal{K}\sigma)$. We remark that equality of patterns do not imply equality of values w.r.t σ since index of constrants may change. However, equality of patterns will be sufficient to guaranty that at indexes s and s', variables validate the same unequalities. Thus, we will "remove" the index s' from σ and show that s can effectively replace s' in any case. That is, with γ replacing any c_v by c_{v-1} if $v > s'$, and replacing any $c_{s'}$ by c_s, $c \in \overrightarrow{\mathcal{C}}$, let σ' be defined as :

- $\forall i < s'$, $\forall\overrightarrow{X} \in \overrightarrow{\mathcal{X}}$, $\sigma'(X_i) = \gamma(\sigma(X_i))$;
- $\forall i \geq s'$, $\forall\overrightarrow{X} \in \overrightarrow{\mathcal{X}}$, $\sigma'(X_i) = \gamma(\sigma(X_{i+1}))$;
- $\forall X \in \mathcal{X}$, $\sigma'(X) = \gamma(\sigma(X))$.

Assuming that $F'' = \forall Q \exists R\, B_1 \vee .. \vee B_p$, let τ'_Q be any index substitution from Q to $\{1..e-1\}$. Thus, let τ_Q be *the* index substitution from Q to $\{1..e\}$ such that :

- $\forall i \in Q$, if $\tau'_Q(i) < s'$ then $\tau_Q(i) = \tau'_Q(i)$, otherwise $\tau_Q(i) = \tau'_Q(i) + 1$.

i.e. we "insert" a blank column at index s'. Now, by definition of σ, $\exists\tau \supseteq \tau_Q$ such that $\sigma \in [\![\, B_1 \vee .. \vee B_p \,]\!]^e_\tau$. Thus, we define $\tau' \supseteq \tau'_Q$ such that :

- $\forall r \in R$, if $\tau(r) < s'$ then $\tau'(r) = \tau(r)$; if $\tau(r) = s'$ then $\tau'(r) = s$; otherwise $\tau'(r) = \tau(r) - 1$.

i.e. we "remove" the column at index s'. We must prove that $\sigma' \in [\![\, B_1 \vee .. \vee B_p \,]\!]^e_{\tau'}$. However, $\exists i \in \{1..p\}$ such that $\sigma \in [\![\, B_i \,]\!]^e_\tau$, thus $\forall ctr \in B_i$, $\sigma \in [\![\, ctr \,]\!]^e_\tau$. As usual, we have different cases depending on ctr :

- If $ctr = (X = u)^{sm}$ or $ctr = (X \in Forge_c(E,\mathcal{K}))$, then $\sigma' \in [\![\, ctr \,]\!]^e_{\tau'}$ since $\sigma'(X) = \sigma(X)$;
- If $ctr = (X_m = u)^\bullet$, then :
  - If $\tau(m) \neq s'$ and $Var^{\mathcal{X}}_{\mathcal{I}}(u) = \{j\}$ with $\tau(j) \neq s'$, then $\sigma'(X_m\tau') = \gamma(\sigma(X_m\tau)) = \gamma(\sigma(u\tau)) = \sigma'(u\tau')$ since any indexed variable in u must have m as index, and $\forall\overrightarrow{Y}$, $\sigma'(Y_m\tau') = \gamma(\sigma(Y_m\tau))$ since $\tau(m) \neq s'$; This includes $j = m$;
  - If $\tau(m) = s'$ and $Var^{\mathcal{X}}_{\mathcal{I}}(u) = \{m\}$, then $\sigma'(X_m\tau') = \gamma(\sigma(X_s))$. Since X_s and $X_{s'}$ share the same pattern for σ, and since σ validates ctr, this pattern is necessarily u, i.e. $\gamma(\sigma(X_s)) = \gamma(\sigma(u[m \leftarrow s])) = \sigma'(u\tau')$;
  - If $\tau(m) = s'$ and $Var^{\mathcal{X}}_{\mathcal{I}}(u) = \{j\} \neq \{m\}$, then as above $\sigma'(X_m\tau') = \gamma(\sigma(X_s))$, and X_s and $X_{s'}$ share the same pattern u. Thus, $\sigma(X_s) = \sigma(u[j \leftarrow \tau(j)])$. Now, either $\tau(j) \neq s'$ and thus $\gamma(\sigma(u\tau)) = \sigma'(u\tau')$, or $\tau(j) = s'$ and thus $\gamma(\sigma(u\tau)) = \sigma'(u[j \leftarrow s]) = \sigma'(u\tau')$. In both cases, $\sigma'(X_s) = \gamma(\sigma(X_s)) = \sigma'(u\tau')$, and thus, $\sigma' \in [\![\, ctr \,]\!]^e_{\tau'}$.
- If $ctr = \left(X_m = t_{\overrightarrow{X}}\right)^\bullet$, then $\sigma'(X_m\tau') = \gamma(\sigma(X_m\tau)) = \gamma(t_{\overrightarrow{X}}) = t_{\overrightarrow{X}}$ since X_s and $X_{s'}$ share the same pattern, and in this case this pattern is $t_{\overrightarrow{X}}$. Thus, $\sigma' \in [\![\, ctr \,]\!]^e_{\tau'}$;
- If $ctr = (\forall k\, X_m \neq u)$, then :
  - If $\tau(m) \neq s'$, then $\forall v \in \{1..e\}$, $\sigma'(X_m\tau') = \gamma(\sigma(X_m\tau)) \neq \gamma(\sigma(u\tau[k \leftarrow v]))$, and thus $\forall v \in \{1..e-1\}$, $\sigma'(X_m\tau') \neq \sigma'(u\tau'[k \leftarrow v])$;
  - If $\tau(m) = s'$, then as above $\forall v \in \{1..e\}$, $\sigma(X_{s'}) = \sigma(X_m\tau) \neq \sigma(u\tau[k \leftarrow v])$. However, X_s and $X_{s'}$ share the same patterns, and for both of them this is the minimal one w.r.t $\lhd$. Thus, u is not the pattern of X_s, i.e. $\forall v \in \{1..e\}$, $\sigma(X_s) \neq \sigma(u\tau[k \leftarrow v])$. It follows that $\forall v \in \{1..e-1\}$, $\sigma'(X_m\tau') \neq \sigma'(u\tau'[k \leftarrow v])$, i.e. $\sigma' \in [\![\, ctr \,]\!]^e_{\tau'}$.
- If $ctr = (X_m \notin Forge_c(E,\mathcal{K}))$, then by assumption on s and s' we know that $\sigma(X_s)$ and $\sigma(X_{s'})$ are both in $\in DY_c(E'\sigma,\mathcal{K}\sigma)$ or none is, for any $(\_ \in Forge_c(E',\mathcal{K}))$ in F'', i.e. including E. Thus, $\sigma'(X_m\tau') \notin DY_c(E\sigma',\mathcal{K}\sigma')$ and $\sigma' \in [\![\, ctr \,]\!]^e_{\tau'}$.

Since these are the only constraints available in F'', it follows that, as expected, $\sigma' \in [\![\, B_i \,]\!]^e_{\tau'}$. Therefore, for any τ'_Q from Q to $\{1..e-1\}$, we could find $\tau' \supseteq \tau'_Q$ from $Q \cup R$ to $\{1..e-1\}$ such that $\sigma' \in [\![\, B_i \,]\!]^e_{\tau'}$, thus proving that $\sigma' \in [\![\, F'' \,]\!]^{e-1}_\emptyset$. This ends the proof of the lemma. □

This claim shows naturally by iteration that $[\![F'']\!]^e_\emptyset$ has a solution with $e > e_{max}$ if and only if $[\![F'']\!]^{e_{max}}_\emptyset$ has a solution. And since for any e, $[\![F'']\!]^e_\emptyset$ has a solution if and only if $[\![F]\!]^e_\emptyset$ has a solution, this proves the property.

□

10 Analysing Asokan-Ginzboorg

We have checked two scenarii of the synchronous version of the Asokan-Ginzboorg protocol (AG protocol). Although this version of the AG protocol has not autonomous keys, the analysis of the two scenarii terminates and gives two results. In the first one, we consider a 'normal' execution of a single session. We show that this execution is secure against a secrecy attack on the group key. In the second scenario, we consider two parallel sessions. We have found an authentication attack. Note that for these scenarii, although the version of the protocol does not only consider autonomous keys, the analysis terminates.

10.1 First Scenario: Asokan-Ginzboorg with a Single Session

Consider a single session of the tagged version of Asokan-Ginzboorg, specified as follows:

$$
\begin{array}{ll}
(L,1) & Init \Rightarrow mpair(t, \langle l, \{e\}_p \rangle) \\
(S,1) & mpair(i, \langle L, \{E_i\}_p \rangle) \Rightarrow mpair(j, (\langle a_j, (\{\langle r_j, s_j \rangle\}_{(E_j)^j})^j \rangle)^j) \\
(L,2) & mpair(k, (\langle a_k, (\{\langle R_k, S_k \rangle\}_{(e)^k})^k \rangle)^k) \Rightarrow mpair(m, (\{\langle mpair(o, (S_o)^o), s' \rangle\}_{(R_m)^m})^m) \\
(S,2) & mpair(q, (\{\langle mpair(u, (s_u)^u), S' \rangle\}_{(r_q)^q})^q) \Rightarrow \\
 & mpair(w, \langle a_w, \{\langle (s_w)^w, H(\langle mpair(y, (s_y)^y), S' \rangle) \rangle\}_{F(\langle mpair(y,(s_y)^y),S' \rangle)} \rangle) \\
(L,3) & mpair(x, \langle a_x, \{\langle (S_x)^x, H(\langle mpair(z, (S_z)^z), s' \rangle) \rangle\}_{F(\langle mpair(z,(S_z)^z),s' \rangle)} \rangle) \Rightarrow End
\end{array}
$$

The ordering on steps is: $W_L = 1, 2, 3$, $W_S = 1, 2$ with $1 <_{W_L} 2$, $2 <_{W_L} 3$, and $1 <_{W_S} 2$. The execution that we focus on is the following: $\langle (L,1), (S,1), (L,2), (S,2), (L,3) \rangle$. The property we want to check is the secrecy of the group key: $Sec = F(\langle mpair(z, (S_z)^z), s' \rangle)$. Then, the constraint system S that we have to solve is specified below:

$$
\begin{array}{ll}
(\text{step } 1) & mpair(i, \langle L, \{E_i\}_p \rangle) \in Forge(E_1, \emptyset) \\
(\text{step } 2) & mpair(k, (\langle a_k, (\{\langle R_k, S_k \rangle\}_{(e)^k})^k \rangle)^k) \in Forge(E_2, \emptyset) \\
(\text{step } 3) & mpair(q, (\{\langle mpair(u, (s_u)^u), S' \rangle\}_{(r_q)^q})^q) \in Forge(E_3, \emptyset) \\
(\text{step } 4) & mpair(x, \langle a_x, \{\langle (S_x)^x, H(\langle mpair(z, (S_z)^z), s' \rangle) \rangle\}_{F(\langle mpair(z,(S_z)^z),s' \rangle)} \rangle) \in Forge(E_4, \emptyset) \\
(\text{step } 5) & Sec \in Forge(E_5, \emptyset)
\end{array}
$$

where E_i are defined as follows:

$$
\begin{array}{lcl}
E_1 & = & \{mpair(t, \langle l, \{e\}_p \rangle)\} \\
E_2 & = & E_1 \cup \{mpair(j, (\langle a_j, (\{\langle r_j, s_j \rangle\}_{(E_j)^j})^j \rangle)^j)\} \\
E_3 & = & E_2 \cup \{mpair(m, (\{\langle mpair(o, (S_o)^o), s' \rangle\}_{(R_m)^m})^m)\} \\
E_4 & = & E_3 \cup \{mpair(w, \langle a_w, \{\langle (s_w)^w, H(\langle mpair(y, (s_y)^y), S' \rangle) \rangle\}_{F(\langle mpair(y,(s_y)^y),S' \rangle)} \rangle)\} \\
E_5 & = & E_4
\end{array}
$$

The normalisation of $S \backslash \{\text{step } 5\}$ by our rules system gives the system below. Note that this system is not put in the disjunctive form to avoid redundancy.

$$
(S \backslash \{\text{step } 5\})\downarrow \;=\; \forall i,\, j,\, k,\, k_2,\, q,\, o,\, q_1,\, o_1,\, x,\, z,\, z_1,\, z_2,\, z_3,\, z_4,\, z_5 \\
(\text{step } 1)\downarrow \,\wedge\, (\text{step } 2)\downarrow \,\wedge\, (\text{step } 3)\downarrow \,\wedge\, (\text{step } 4)\downarrow
$$

where,

$$
(\text{step } 1)\downarrow \;=\; \left| \begin{array}{l}
((L \in Forge_c(E_1, \emptyset) \wedge (E_i = e)^m) \vee ((L = mpair(t, \langle l, \{e\}_p \rangle))^{sm} \wedge (E_i = e)^m) \\
\vee ((L = \langle l, \{e\}_p \rangle)^{sm} \wedge (E_i = e)^m) \vee ((L = l)^{sm} \wedge (E_i = e)^m) \\
\vee ((L = \{e\}_p)^{sm} \wedge (E_i = e)^m) \vee ((L = l)^{sm} \wedge (E_j = e)^m))
\end{array} \right.
$$

$$(\text{step } 2)\downarrow = \left| \begin{array}{l} (((R_k = r_k)^m \wedge (S_k = s_k)^m \wedge (E_k = e)^f) \\ \vee((R_{k_2} = r_{k_2})^m \wedge (S_{k_2} = s_{k_2})^m \wedge (E_{k_2} = e)^f)) \end{array} \right.$$

$$(\text{step } 3)\downarrow = \left| \begin{array}{l} (((R_q = r_q)^f \wedge (S' = s')^{sm} \wedge (S_o = s_o)^f) \\ \vee((R_{q_1} = r_{q_1})^f \wedge (S' = s')^{sm} \wedge (S_{o_1} = s_{o_1})^f)) \end{array} \right.$$

$$(\text{step } 4)\downarrow = \left| \begin{array}{l} (((S_x = s_x)^f \wedge (S_{z_3} = s_{z_3})^f \wedge (S_{z_3} = s_{z_3})^f) \vee \\ ((S_x = s_x)^f \wedge (S_{z_1} = s_{z_1})^f \wedge (S_z = s_z)^f) \vee \\ ((S_{x_1} = s_{x_1})^f \wedge (S_{z_5} = s_{z_5})^f \wedge (S_{z_2} = s_{z_2})^f)) \end{array} \right.$$

If we try to normalize the last step step 5 leads to :

$$\begin{array}{l} \exists m\ \exists o_2\ \exists w \\ (F(\langle mpair(z,(S_z)^z), s'\rangle) \in Sub_d(S', E_5, \mathcal{E}, \emptyset) \\ \wedge F(\langle mpair(y,(s_y)^y), S'\rangle) \in Forge(E_5, \{\{\langle (s_w)^w, H(\langle mpair(y,(s_y)^y), S'\rangle)\rangle\}_{F(\langle mpair(y,(s_y)^y), S'\rangle)}\})) \\ \vee(F(\langle mpair(z,(S_z)^z), s'\rangle) \in Sub_d((S_{o_2})^{o_2}, E_5, \mathcal{E}, \emptyset) \wedge \\ \qquad (R_m)^m \in Forge(E_5, \{\{\langle mpair(o,(S_o)^o), s'\rangle\}_{(R_m)^m}\})) \end{array}$$

The interleaving between $F(\langle mpair(z,(S_z)^z), s'\rangle) \in Sub_d(S', E_5, \mathcal{E}, \emptyset)$ with each block obtained above leads to $\perp$ since in each block, the submaster constraint for S' is $S' = s'$. In the same way, the interleaving between $F(\langle mpair(z,(S_z)^z), s'\rangle) \in Sub_d((S_{o_2})^{o_2}, E_5, \mathcal{E}, \emptyset)$ and the master constraint for S_o in each block leads to $\perp$ since the "value" of S_o is a constant s_o. Thus, we conclude that an execution of Asokan-Ginzboorg with a single session is secure against a secrecy attack.

10.2 Second Scenario: Asokan-Ginzboorg with Two Parallel Sessions

Consider two parallel sessions. In the two following specifications (A, i, j) denotes the step i of the participant A in the session j. $(L, \_, 1)$ (resp $(L, \_, 2)$) denotes the leader of the first session (resp second session) and $(S, \_, 1)$ (resp $(S, \_, 2)$) denotes the similator of the first session (resp second session). The specification of the first session is given below:

$$\begin{array}{ll} (L,1,1) & Init \Rightarrow mpair(t, \langle l, \{e\}_p \rangle) \\ (S,1,1) & mpair(i, \langle L, \{E_i\}_p \rangle) \Rightarrow mpair(j, (\langle a_j, (\{\langle r_j, s_j \rangle\}_{(E_j)^j})^j \rangle)^j) \\ (L,2,1) & mpair(k, (\langle a_k, (\{\langle R_k, S_k \rangle\}_{(e)^k})^k \rangle)^k) \Rightarrow mpair(m, (\{\langle mpair(o,(S_o)^o), s' \rangle\}_{(R_m)^m})^m) \\ & \wedge \text{witness}((L,\_,1),(S,\_,1), f\_s, \langle mpair(o,(S_o)^o), s' \rangle) \\ (S,2,1) & mpair(q, (\{\langle mpair(u,(s_u)^u), S' \rangle\}_{(r_q)^q})^q) \Rightarrow \\ & mpair(w, \langle a_w, \{\langle (s_w)^w, H(\langle mpair(y,(s_y)^y), S' \rangle)\rangle\}_{F(\langle mpair(y,(s_y)^y), S' \rangle)} \rangle) \\ & \wedge \text{request}((S,\_,1),(L,\_,1), f\_s, \langle mpair(u,(s_u)^u), S' \rangle) \\ (L,3,1) & mpair(x, \langle a_x, \{\langle (S_x)^x, H(\langle mpair(z,(S_z)^z), s' \rangle)\rangle\}_{F(\langle mpair(z,(S_z)^z), s' \rangle)} \rangle) \Rightarrow End \end{array}$$

The specification of the second session is given below:

$$\begin{array}{ll} (L,1,2) & Init \Rightarrow mpair(t, \langle l, \{e2\}_p \rangle) \\ (S,1,2) & mpair(i, \langle L', \{E'_i\}_p \rangle) \Rightarrow mpair(j, (\langle a_j, (\{\langle r'_j, s'_j \rangle\}_{(E'_j)^j})^j \rangle)^j) \\ (L,2,2) & mpair(k, (\langle a_k, (\{\langle R'_k, S'_k \rangle\}_{(e2)^k})^k \rangle)^k) \Rightarrow mpair(m, (\{\langle mpair(o,(S'_o)^o), s" \rangle\}_{(R'_m)^m})^m) \\ & \wedge \text{witness}((L,\_,2),(S,\_,2), s\_s, \langle mpair(o,(S'_o)^o), s" \rangle) \\ (S,2,2) & mpair(q, (\{\langle mpair(u,(s'_u)^u), S" \rangle\}_{(r'_q)^q})^q) \Rightarrow \\ & mpair(w, \langle a_w, \{\langle (s'_w)^w, H(\langle mpair(y,(s'_y)^y), S" \rangle)\rangle\}_{F(\langle mpair(y,(s'_y)^y), S" \rangle)} \rangle) \\ & \wedge \text{request}((S,\_,2),(L,\_,2), s\_s, \langle mpair(u,(s'_u)^u), S" \rangle) \\ (L,3,2) & mpair(x, \langle a_x, \{\langle (S'_x)^x, H(\langle mpair(z,(S'_z)^z), s" \rangle)\rangle\}_{F(\langle mpair(z,(S'_z)^z), s" \rangle)} \rangle) \Rightarrow End \end{array}$$

Steps are ordered as follow: $W_L = 1, 2, 3$, $W_S = 1, 2$ with $1 <_{W_L} 2$, $2 <_{W_L} 3$, and $1 <_{W_S} 2$ for each session. We consider the following execution:
$(L,1,1),(L,1,2),(S,1,1),(S,1,2),(L,2,1),(L,2,2),(S,2,1),(S,2,2),(L,3,1),(S,3,2)$. The property we want to check is the authentication of the group key in each session which is defined in the two specification below

by *witness* and *request*. Then, the constraint system S that we have to solve is specified below:

$$
\begin{array}{ll}
(\text{step } 1) & mpair(i, \langle L, \{E_i\}_p \rangle) \in Forge(E_2, \emptyset) \\
(\text{step } 2) & mpair(i, \langle L', \{E'_i\}_p \rangle) \in Forge(E_3, \emptyset) \\
(\text{step } 3) & mpair(k, (\langle a_k, (\{\langle R_k, S_k \rangle\}_{(e)^k})^k \rangle)^k) \in Forge(E_4, \emptyset) \\
(\text{step } 4) & mpair(k, (\langle a_k, (\{\langle R'_k, S'_k \rangle\}_{(e2)^k})^k \rangle)^k) \in Forge(E_5, \emptyset) \\
(\text{step } 5) & mpair(q, (\{\langle mpair(u, (s_u)^u), S' \rangle\}_{(r_q)^q})^q) \in Forge(E_6, \emptyset) \\
(\text{step } 6) & mpair(q, (\{\langle mpair(u, (s'_u)^u), S" \rangle\}_{(r'_q)^q})^q) \in Forge(E_7, \emptyset) \\
(\text{step } 7) & mpair(x, \langle a_x, \{\langle (S_x)^x, H(\langle mpair(z, (S_z)^z), s' \rangle) \rangle\}_{F(\langle mpair(z,(S_z)^z), s' \rangle)} \rangle) \in Forge(E_8, \emptyset) \\
(\text{step } 8) & mpair(x, \langle a_x, \{\langle (S'_x)^x, H(\langle mpair(z, (S'_z)^z), s" \rangle) \rangle\}_{F(\langle mpair(z,(S'_z)^z), s" \rangle)} \rangle) \in Forge(E_9, \emptyset)
\end{array}
$$

where E_i are defined as follows:

$$
\begin{array}{lcl}
E_2 & = & \{mpair(t, \langle l, \{e\}_p \rangle), mpair(t, \langle l2, \{e2\}_p \rangle)\} \\
E_3 & = & E_2 \cup \{mpair(j, (\langle a_j, (\{\langle r_j, s_j \rangle\}_{(E_j)^j})^j \rangle)^j)\} \\
E_4 & = & E_3 \cup \{mpair(j, (\langle a_j, (\{\langle r'_j, s'_j \rangle\}_{(E'_j)^j})^j \rangle)^j)\} \\
E_5 & = & E_4 \cup \{mpair(m, (\{\langle mpair(o, (S_o)^o), s' \rangle\}_{(R_m)^m})^m)\} \\
E_6 & = & E_5 \cup \{mpair(m, (\{\langle mpair(o, (S'_o)^o), s" \rangle\}_{(R'_m)^m})^m)\} \\
E_7 & = & E_6 \cup \{mpair(w, \langle a_w, \{\langle (s_w)^w, H(\langle mpair(y, (s_y)^y), S' \rangle) \rangle\}_{F(\langle mpair(y,(s_y)^y), S' \rangle)} \rangle)\} \\
E_8 & = & E_7 \cup \{mpair(w, \langle a_w, \{\langle (s'_w)^w, H(\langle mpair(y, (s'_y)^y), S" \rangle) \rangle\}_{F(\langle mpair(y,(s'_y)^y), S" \rangle)} \rangle)\} \\
E_9 & = & E_8
\end{array}
$$

We normalise S by our rules system. We just focus on a single block of the normalized system:

$$
\begin{array}{l}
(L \in Forge_c(E_2, \emptyset) \wedge L' \in Forge_c(E_3, \emptyset) \wedge (E_{i_1} = e2)^m \wedge (E'_{i_4} = e)^m \wedge \\
(R_{k_1} = r'_{k_1})^m \wedge (S_{k_1} = s'_{k_1})^m \wedge (E'_{j_1} = e)^f \wedge \\
(R'_{k_2} = r_{k_2})^f \wedge (S'_{k_2} = s_{k_2})^f \wedge (E_{j_2} = e2)^f \wedge \\
(R'_{q_1} = r_{q_1})^f \wedge (S' = s")^{sm} \wedge (S'_{o_1} = s_{o_1})^f \wedge \\
(R_{q_2} = r'_{q_2})^f \wedge (S" = s')^{sm} \wedge (S_{o_2} = s'_{o_2})^f \wedge \\
(S_{x_1} = s'_{x_1})^f \wedge (S_{y_1} = s'_{y_1})^f \wedge (S'_{x_2} = s_{x_2})^f \wedge (S'_{z_1} = s_{z_1})^f \\
\wedge \text{witness}((L, \_, 1), (S, \_, 1), f\_s, \langle mpair(o, (S_o)^o), s' \rangle) \\
\wedge \text{witness}((L, \_, 2), (S, \_, 2), s\_s, \langle mpair(o, (S'_o)^o), s" \rangle) \\
\wedge \text{request}((S, \_, 1), (L, \_, 1), f\_s, \langle mpair(u, (s_u)^u), S' \rangle) \\
\wedge \text{request}((S, \_, 2), (L, \_, 2), s\_s, \langle mpair(u, (s'_u)^u), S" \rangle))
\end{array}
$$

Index variables in this block are quantified in S by:

$$\forall i_1, \forall i_4, \forall k_1, \forall k_2, \forall q_1, \forall q_2, \forall o_1, \forall o_2, \forall x_1, \forall x_2, \forall y_1, \forall z_1, \ \ \exists j_1, \exists j_2$$

This block corresponds to an authentication attack on f_s. Indeed, the solution σ of this block gives the value s'_o to S_o. Thus, the *request* for f_s gives a value different from the one of the corresponding *witness*. The same raison is valid for s_s.

11 Conclusion and Further Works

We have proposed an extension of the constraint-based approach in symbolic protocol verification in order to handle a class of protocols (the Well-Tagged ones with Autonomous Keys) which admit unbounded lists in messages. This class can be used to model in particular interesting group protocols. We have proposed a decision procedure for Well-Tagged protocols with Autonomous Keys. We have studied the Asokan-Ginzboorg protocol with keys having index variables outside *mpair*'s. We showed that the analysis of this protocol terminates. Thus, we conjecture termination for a larger class than Well-Tagged protocols where the restriction of autonomous keys is weakened to include at least this protocol.

References

[AC02] A. Armando and L. Compagna. Automatic SAT-Compilation of Protocol Insecurity Problems via Reduction to Planning. In *Foundation of Computer Security & Verification Workshops*, Copenhagen, Denmark, July 25-26 2002.

[AG00] N. Asokan and P. Ginzboorg. Key agreement in ad hoc networks. *Computer Communications*, 23(17):1627–1637, 2000.

[BMV03] D. Basin, S. Mödersheim, and L. Viganò. An On-The-Fly Model-Checker for Security Protocol Analysis. In Einar Snekkenes and Dieter Gollmann, editors, *Proceedings of ESORICS'03*, LNCS 2808, pages 253–270. Springer-Verlag, 2003.

[BP03] Bruno Blanchet and Andreas Podelski. Verification of cryptographic protocols: Tagging enforces termination. In Andrew D. Gordon, editor, *Foundations of Software Science and Computational Structures, 6th International Conference, FOSSACS 2003 Held as Part of the Joint European Conference on Theory and Practice of Software, ETAPS 2003, Warsaw, Poland, April 7-11, 2003, Proceedings*, volume 2620 of *Lecture Notes in Computer Science*, pages 136–152. Springer, 2003.

[DY83] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Trans. Inform Theory IT-29*, pages 198–208, 1983. Also STAN-CS-81-854, May 1981, Stanford U.

[JD97] J.A.Bull and D.J.Otway. The authentication protocol. Technical report, Defence Research Agency, Mavern,UK, 1997.

[KMT08] S. Kremer, A. Mercier, and R. Treinen. Proving group protocols secure against eavesdroppers. In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *Proceedings of the 4th International Joint Conference on Automated Reasoning (IJCAR'08)*, Lecture Notes in Artificial Intelligence, Sydney, Australia, August 2008. Springer-Verlag. To appear.

[KRT07] K.O.Kürtz, R.Küsters, and T.Wilke. Selecting theories and nonce generation for recursive protocols. In *FMSE '07: Proceedings of the 2007 ACM workshop on Formal methods in security engineering*, pages 61–70, New York, NY, USA, 2007. ACM.

[KT07] R. Küsters and T. Truderung. On the automatic analysis of recursive security protocols with xor. Technical report, ETH Zurich, 2007. An abridged version appears in STACS 2007.

[KW04] R. Küsters and T. Wilke. Automata-based analysis of recursive cryptographic protocols. In *21st Symposium on Theoretical Aspects of Computer Science (STACS 2004)*, Lecture Notes in Computer Science. Springer-Verlag, 2004.

[Mea00] C. Meadows. Extending formal cryptographic protocol analysis techniques for group protocols and low-level cryptographic primitives. In P.Degano, editor, *the First Workshop on Issues in the Theory of Security*, pages 87–92, Geneva, Switzerland, July 2000.

[Mit97] S. Mittra. Iolus: A framework for scalable secure multicasting. In *SIGCOMM*, pages 277–288, 1997.

[MS01] C. Meadows and P. Syverson. Formalizing gdoi group key management requirements in npatrl. In *CCS'01: Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 235–244, New York, USA, 2001. ACM Press.

[MT07] José Meseguer and Prasanna Thati. Symbolic reachability analysis using narrowing and its application to verification of cryptographic protocols. *Higher-Order and Symbolic Computation*, 20(1-2):123–160, 2007.

[Pau96] L. C. Paulson. Isabelle: A generic theorem prover. *Lecture Notes in Computer Science*, 828:283–298, 1996.

[Pau97] L. C. Paulson. Mechanized proofs for a recursive authentication protocol. In *10th Computer Security Foundations Workshop*, pages 84–95. IEEE Computer Society Press, 1997.

[PQ03] O. Pereira and J.-J. Quisquater. Some attacks upon authenticated group key agreement protocols. *Journal of Computer Security*, 11(4):555–580, 2003.

[PQ04] O. Pereira and J.-J. Quisquater. Generic insecurity of cliques-type authenticated group key agreement protocols. In *CSFW*, pages 16–19, 2004.

[RS03] Ramaswamy Ramanujam and S. P. Suresh. Tagging makes secrecy decidable with unbounded nonces as well. In Paritosh K. Pandya and Jaikumar Radhakrishnan, editors, *FST TCS 2003: Foundations of Software Technology and Theoretical Computer Science, 23rd Conference, Mumbai, India, December 15-17, 2003, Proceedings*, volume 2914 of *Lecture Notes in Computer Science*, pages 363–374. Springer, 2003.

[RT03] M. Rusinowitch and M. Turuani. Protocol insecurity with a finite number of sessions, composed keys is np-complete. *Theor. Comput. Sci.*, 1-3(299):451–475, 2003.

[SB04] G. Steel and A. Bundy. Attacking group multicast key management protocols using CORAL. In A. Armando and L. Viganó, editors, *Proceedings of the ARSPA Workshop*, volume 125 of *ENTCS*, pages 125–144, 2004.

[SWT98] M. Steiner, M. Waidner, and G. Tsudik. Cliques: A new approach to group key agreement. In *ICDCS '98: Proceedings of the The 18th International Conference on Distributed Computing Systems*, page 380, Washington, DC, USA, 1998. IEEE Computer Society.

[TJ03] M. Taghdiri and D. Jackson. A lightweight formal analysis of a multicast key management scheme. In *FORTE*, pages 240–256, 2003.

[Tru05] T. Truderung. Selecting theories and recursive protocols. pages 217–232, 2005.

[Tur06] M. Turuani. The CL-Atse Protocol Analyser. In *Term Rewriting and Applications - Proc. of RTA*, volume 4098 of *Lecture Notes in Computer Science*, pages 277–286, Seattle, WA, USA, 2006.

[Wei99] C. Weidenbach. Towards an automatic analysis of security protocols in first-order logic. In *16th International Conference on Automated Deduction*, volume 1632, pages 314–328. Springer, 1999.

Centre de recherche INRIA Nancy – Grand Est
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399